**SQL\*LOADER**

**SQL\*Loader** - это продукт для загрузки данных из внешних файлов в таблицы БД ORACLE. SQL\*Loader загружает данные в различных форматах, выполняет фильтрацию (выборочную загрузку записей в зависимости от значений данных), и может загружать несколько таблиц одновременно. Во время выполнения SQL\*Loader формирует детальный файл отчета (Log File) со статистикой загрузки и может также создавать файл отвергнутых (Discard File) записей (записи, отвергнутые из-за ошибок в данных) и файл пропущенных записей (Bad File) (записи, которые не соответствуют критерию выбора).

Вы можете управлять загрузкой при помощи нескольких опций. SQL\*Loader может:

– загружать данные из нескольких файлов данных различных типов;

– обрабатывать записи фиксированного формата, свободного формата, а также записи переменной длины;

– поддерживать широкий спектр типов данных;

– загружать несколько таблиц за один прогон; загружать выбранные строки в каждую таблицу;

– комбинировать несколько физических записей в одну логическую запись;

– обрабатывать одну физическую запись как несколько логических записей;

– генерировать уникальное последовательное значение ключа в указанной колонке;

– использовать систему управления файлами или записями вашей ОС для доступа к файлам данных;

– загружать данные с диска;

– обеспечить полный контроль и диагностику ошибок, что позволяет легко выверить и загрузить все данные;

– использовать управляющий файл, совместимый с файлом загрузчика DB2 Корпорации IBM.

**ВХОДНАЯ ИНФОРМАЦИЯ SQL\*LOADER**

Чтобы загрузить данные из внешних файлов в БД ORACLE, вы должны подготовить для SQL\*Loader входную информацию двух типов: сами данные и управляющую информацию, описывающую, как выполнять загрузку. Данные, загружаемые в БД ORACLE, должны находиться в файлах на диске. Эти файлы данных SQL\*Loader должны интерпретироваться некоторым образом при загрузке. Информация для этого находится в управляющем файле.

**ПОДГОТОВКА БАЗЫ ДАННЫХ**

Таблицы, в которые загружаются данные, должны уже существовать в БД ORACLE. Нет специальных требований к этим таблицам; они могут быть кластерными или индексированными, они могут быть в действительности представлениями (view), для которых установлены полномочия INSERT. Заметьте, что загрузка обычно выполняется быстрее, если индексы создаются после загрузки.

Таблицы могут уже иметь данные или быть пустыми.

**ВЫХОДНАЯ ИНФОРМАЦИЯ SQL\*LOADER**

Когда SQL\*Loader загружает данные, он может создавать до трех файлов, как было описано выше.

Определим назначение каждого из них подробнее.

**Файл отчета (LOG FILE)**

Когда SQL\*Loader начинает выполняться, он создает файл отчета. Если невозможно создать этот файл, выполнение прекращается. Файл отчета содержит детальную сводку выполнения, состоящую из нескольких опций, как описано ниже:

*Заголовок:*

– дата прогона

– N версии программного продукта

*Глобальная информация:*

– имена всех входных/выходных файлов

– эхо аргументов командной строки

– информация о символе продолжения

*Информация таблицы:*

повторяется для каждой загружаемой таблицы:

– имя

– условия загрузки, если имеются (или все записи, или предложение WHEN, используемое как критерий загрузки)

– одно из предложений INSERT, APPEND, REPLACE

– информация колонок:

– позиция, длина, тип данных и ограничитель колонки, если она имеется в файле данных

* RECNUM, SEQUENCE или CONSTANT, если указано

– DEFAULTIF или NULLIF, если указано

*Информация файла данных*

– помещается в отчет только для тех файлов данных, которые имеют ошибки данных:

– SQL\*Loader/ORACLE ошибки для записей с ошибочными данными;

– пропущенные записи.

*Информация о загрузке таблицы:*

– повторяется для каждой загружаемой таблицы;

– количество загруженных строк;

– количество строк, которые должны были загружаться, но отброшены из-за ошибок данных;

– количество пропущенных строк, не удовлетворяющих условию WHEN;

– количество строк, для которых все используемые поля были пустыми;

*Итоговая статистика:*

– используемое пространство;

– для двоичных массивов (действительно используемое по сравнению с указанным в BINDSIZE);

– для другой служебной информации (требуется всегда и не зависит от BINDSIZE);

– Итоговая статистика загрузки. Для каждого файла данных содержит количество записей, которые были

– пропущены до обработки;

– прочитаны;

– отброшены;

– пропущены при обработке;

– время начала и окончания прогона;

– полное время выполнения;

– полное время процессора (включает время всех операций ввода-вывода, но может не включать фоновое время процессора ORACLE).

Большая часть этой информации выдается при успешном выполнении SQL\*Loader. Однако обнаруженные ошибки приводят к выдаче различных порций отчета. Например, если сам файл отчета не может быть создан, выполнение SQL\*Loader прекращается. Ошибки, обнаруженные при обработке управляющего файла, будут отмечены в файле отчета, обнаруженные в командной строке - нет.

**Файл отброшенных записей (BAD FILE)**

При выполнении SQL\*Loader создает файл отброшенных записей. Этот файл содержит все записи, которые SQL\*Loader пытался загрузить, но не смог. Эти записи называются отброшенными записями.

Запись отбрасывается, если:

– при загрузке записи возникает ошибка ORACLE (например, неверные данные для некоторого типа данных, дублируется значение в уникальном индексе, пустое значение в поле NOT NULL);

– SQL\*Loader не может определить, правильные данные или нет. Так, невозможно определить, удовлетворяет ли запись условию WHEN в случае несбалансированных ограничителей. Например, если используются ограничители - двойные кавычки, то SQL\*Loader пропустит запись.

Если же условие WHEN может быть проверено даже в случае несбалансированных ограничителей, данные будут соответственно либо загружены, либо отброшены.

Если запись отброшена при загрузке, то никакая часть этой записи не будет загружена в какую-либо таблицу. Например, если данные из одной записи должно быть вставлены в несколько таблиц, несколько операций выполнилось успешно, а одна завершилась аварийно, то для всех операций вставки из этой записи будет выполнен откат. Запись помещается в файл отброшенных записей, где она может быть исправлена и при необходимости загружена повторно. Предыдущие операции вставки из записи, не содержащей ошибок, не имеют последствий. Файл отчета содержит сообщения об ошибках ORACLE для каждой отброшенной записи. SQL\*Loader всегда создает файл отброшенных записей. Он может быть пустым, если ни одной записи не отброшено. Он записывается в том же формате, что и файл данных, поэтому отброшенные данные после редактирования или необходимого исправления может быть загружены с тем же управляющим файлом.

**Файл пропущенных записей (DISCARD FILE)**

При выполнении SQL\*Loader может создавать файл пропущенных записей. Этот файл создается, если вы укажете одну из 2 опций в командной строке:

DISCARD = filename

DISCARDMAX = nn

или если вы используете одну из опций в управляющем файле:

DISCARDFILE filename

DISCARDDN filename

DISCARDS n

DISCARDMAX n

Итак, файл пропущенных записей создается, если вы явно потребуете это (указав имя файла), или вы неявно потребуете, указав, например, максимальное количество записей для него. В последнем случае имя файла пропущенных записей совпадает с именем файла данных, но принимается расширение имени файла DSC.

Записи помещаются в файл пропущенных записей, если:

– ни одно условие WHEN не выполняется для какой-либо загружаемой таблицы;

– условие WHEN выполняется для таблицы, но все поля пустые, либо из-за NULLIF, либо как отсутствующие.

Если данные из записи загружены в таблицу, эта запись не помещается в файл пропущенных записей. Файл пропущенных записей записывается в том же формате, что и файл данных, поэтому пропущенные данные после редактирования и исправления может быть загружены с тем же управляющим файлом.

**КОМАНДА SQLLDR**

SQL\*Loader вызывается командой ***SQLLDR***, за которой следуют различные аргументы. Если вы вызываете ***SQLLDR*** без аргументов, вы увидите сводку аргументов с кратким описанием и значениями по умолчанию. Некоторые умолчания зависят от OC и такой вызов позволяет увидеть умолчания, используемые в вашей OC. Пример такой сводки приведен ниже. В этом примере приведены значения аргументов по умолчанию, не зависящие от ОС.

   ¦   SQLLDR

   ¦   Usage: SQLLDR keyword=value [, keyword=value, ]

   ¦   Использование:

   ¦          SQLLDR ключ. слово=значение [, ключ. слово=знач.,]

   ¦   Правильные ключевые слова:

   ¦       userid -- имя пользователя ORACLE/пароль

   ¦      control -- имя управляющего файла

   ¦          log -- имя файла отчета

   ¦          bad -- имя файла отброшенных записей

   ¦         data -- имя файла данных

   ¦      discard -- имя файла пропущенных записей

   ¦   discardmax -- допустимое число пропущенных записей

   ¦                 (по умолчанию все)

   ¦         skip -- число логических записей, пропущенных

   ¦                 до обработки (по умолчанию 0)

   ¦         load -- число загружаемых записей (по умолчанию все)

   ¦        rrors -- допустимое число ошибок (по умолчанию 50)

   ¦         rows -- число строк в связанном массиве

   ¦                 (по умолчанию 64)

   ¦     bindsize -- размер связанного массива в байтах

   ¦       silent -- подавление сообщений при выполнении

   ¦                 (заголовок, ошибки, пропущенные записи)

Аргументы следуют за командой и необязательно разделяются запятыми. Они могут вводиться **2 способами**:

 – в любом порядке с использованием ключевых слов для идентификации аргументов

 – в определенном порядке, без ключевых слов.

Например, следующие 2 **командные строки** полностью эквивалентны (утилита SQLLDR вызывается из командной строки операционной системы с перечисленными параметрами):

SQLLDR control=foo.ctl, log=bar.log, bad=baz.bad, data=etc.dat, userid=student/stud@DBSTUD, errors=999, load=2000, discard=toss.dis, discardmax=5

SQLLDR student/stud@DBSTUD, foo.ctl, bar.log, baz.bad, etc.dat, toss.dis, 5, 0, 2000, 999

**USERID:**     идентификация пользователя.

                  Имя пользователя ORACLE и пароль. Если опущены, выдается подсказка. Если указан только слеш  ("/"), умолчание OPS$. Для загрузки в удаленную БД должно быть указана строка спецификации БД SQL\*Net.

**CONTROL:**    управляющий файл.

                  Файл, описывающий, как загружать данные. Если расширение имени файла или тип файла не указаны, используется CTL. Если аргумент опущен, выдается подсказка.

**LOG:**        файл отчета.

                  Файл создается SQL\*Loader и содержит информацию о процессе загрузки. Если имя файла не указано, оно принимается по умолчанию. Если расширение имени или тип файла не указаны, используется LOG.

**BAD:**        файл отброшенных записей.

                  Файл создается SQL\*Loader и содержит записи, в которых обнаружены ошибки при загрузке или нарушен формат данных. Если имя файла не указано, используется имя управляющего файла. Если расширение имени или имя файла не указаны, используется BAD. Этот файл имеет тот же формат, что и файл данных, поэтому он может быть повторно загружен с тем же управляющим файлом после того, как выполнены необходимые корректировки.

**DATA:**       файл данных.

                  Этот файл содержит загружаемые данные. Если имя файла не указано, оно принимается по умолчанию. Если расширение имени или тип файла не указаны, используется DAT. Данные может быть также включены непосредственно в управляющий файл.

**DIRECT** = ( TRUE или FALSE ) –Прямой (direct) режим загрузки данных.

В обычном режиме загрузки данных используются SQL -операторы INSERT . При этом каждая команда выполняется, проходя всю логику SQL -анализатора. Для улучшения производительности загрузки данных (т.е. для ускорения этого процесса) используют дополнительный режим, называемый «прямым» ( direct path ).При использовании прямого режима, данные записываются в файлы данных БД напрямую, минуя SQL -логику. Этот режим быстрее и намного эффективнее, но имеет несколько ограничений. Вот некоторые из ситуаций, когда нельзя применять прямой режим загрузки:

– загружаемая таблица - кластерная;

– с таблицей, в которую происходит загрузка данных, работают посторонние транзакции;

– таблица, в которую направляются данные, индексирована.

**DISCARD:**    файл пропущенных записей.

                  Необязательный файл, который создается, если указан аргумент DISCARD или DISCARDMAX. В этот файл заносятся записи, которые ни отброшены, ни вставлены в загружаемую таблицу. Если имя файла не указано, по умолчанию принимается имя управляющего файла. Если расширение имени или тип файла не указаны, используется DSC. Этот файл имеет тот же формат, что и файл данных, поэтому он может быть повторно загружен с тем же управляющим файлом                  после того, как выполнены необходимые корректировки.

**DISCARDMAX:** допустимое число пропущенных записей.

                  Указывает, сколько записей допускается пропустить, прежде чем загрузка будет прервана. По умолчанию все. Чтобы прервать загрузку на первой пропущенной записи, нужно указать ноль (0).

**SKIP:**       число записей, пропущенных до обработки.

                  Указывает число логических записей с начала файла, которые нужно пропустить перед загрузкой. По умолчанию 0.

**LOAD:**       число загружаемых записей.

                  Максимальное число загружаемых логических записей (не считая записей, указанных аргументом SKIP). По умолчанию все записи должно быть загружены. Если файл данных содержит меньше записей, ошибка не сигнализируется.

**ERRORS:**     допустимое число ошибок.

                  Указывает, сколько ошибок допускается, прежде чем загрузка будет прервана. По умолчанию 50. Чтобы прервать загрузку на первой ошибке, нужно указать ноль (0). Чтобы разрешить все ошибки, нужно указать достаточно большое число.

**ROWS:**       число строк в связанном массиве.

                  Число строк в буфере для выполнения нескольких включений за одну операцию COMMIT. По умолчанию 64

**BINDSIZE:**   максимальный размер в байтах.

                  Максимальный размер связанного массива в байтах. Этот размер отвергает значение по умолчанию (зависящее от ОС), а также значение, вычисленное на основе ROW.

**SILENT:**       подавление.

Используйте соответствующие ключевые слова для подавления:

**HEADER:**     заголовок SQL\*Loader.

**FEEDBACK:**   Feedback в каждой точке COMMIT.

**ERRORS:**    сообщений в файле отчета о каждой записи, вызывающей ошибку ORACLE (число отброшенных записей по прежнему будет указано).

**DISCARDS**:   сообщений в файле отчета о каждой записи, помещенной в файл пропущенных записей.

**ALL:**        Выполняет все перечисленное выше.

**СИНТАКСИС УПРАВЛЯЮЩЕГО ФАЙЛА**

**Создание управляющего файла**

Управляющий файл содержит следующую информацию:

 – где находятся загружаемые данные. Указывается либо имя файла (файлов) данных в операторе INFILE, либо начало данных в управляющем файле оператором BEGINDATA. Если ни INFILE, ни BEGINDATA не используются, имя файла данных определяется по аргументам командной строки.

 – в какие таблицы БД должно быть загружены; используется оператор INTO TABLE

 – соответствие между полями файла данных и колонками таблиц БД, как часть оператора LOAD DATA. Общие правила подготовки управляющего файла следующие:

   Файл готовится в свободном формате. Так, оператор может продолжаться со строки на строку и занимать несколько строк, начинающихся с любого слова

    Файл готовится на верхнем или нижнем регистре. Регистр имеет значение только для строк, заключенных в кавычки

    Допускаются комментарии в любой строке. Они начинаются двойным тире. SQL\*Loader игнорирует любой текст от двойного тире до конца строки. Однако комментарии не могут появляться в файле данных или области данных управляющего файла. В этих случаях двойное тире рассматривается как часть данных.

   Ключевое слово SQL\*Loader может быть указано в кавычках, если вы желаете использовать его как имя таблицы или колонки.

**СИНТАКСИС ОПЕРАТОРА LOAD DATA**

Управляющий файл обычно начинается фразой LOAD DATA, за которой следуют несколько фраз, описывающих загружаемые данные. (Только комментарии или оператор OPTIONS может предшествовать оператору LOAD DATA).

LOAD DATA**:** полный синтаксис управляющего оператора:

   ¦  OPTIONS  (options)

   ¦

   ¦  LOAD [DATA]

   ¦  [ { INFILE | INDDN } { filename | \* }

   ¦    [ STREAM | RECORD | FIXED len

   ¦      [ BLOCKSIZE size ] | VARIABLE [len] ] ]

   ¦  [ { BADFILE | BADDN } filename ]

   ¦  [ { DISCARDFILE | DISCARDDN } filename ]

   ¦  [ { DISCARDS | DISCARDMAX } n ]

   ¦

   ¦  [ { INDDN | INFILE } ]

   ¦

   ¦  [ APPEND | REPLACE | INSERT ]

   ¦  [ RECLEN n ]

   ¦  [ { CONCATENATE n | CONTINUEIF

   ¦      { [THIS | NEXT] (start[:end]) | LAST }

   ¦               operator {'char\_str' | X'hex\_str'} } ]

   ¦

   ¦  INTO TABLE tablename

   ¦      [ APPEND | REPLACE | INSERT ]

   ¦      [ WHEN field\_condition [ AND field\_condition ]]

   ¦      [ FIELDS [delimiter\_spec] ]

   ¦     (

   ¦       column\_name

   ¦       {

   ¦         RECNUM |

   ¦         CONSTANT value |

   ¦         SEQUENCE ( { n | MAX | COUNT } [ , increment ] ) |

   ¦         [ POSITION ( { start [:end] | \* [+n] })] datatype\_spec

   ¦           [ NULLIF field\_condition ]

   ¦           [ DEFAULTIF field\_condition ]

   ¦       }

   ¦       [ , ]

   ¦     )

   ¦  [ INTO TABLE ]

   ¦

   ¦  [ BEGINDATA ]

Далее описаны некоторые операторы из управляющего файла. Полное описание синтаксиса управляющего файла, а также использование его опций вы можете найти в соответствующей документации ORACLE SQL\*LOADER.

DELIMITER\_ESPEC

Ограничители указываются в следующем формате:

                  [ TERMINATED [ BY ] { WHITESPACE | [X] 'char' } ]

                  [ [ OPTIONALLY ]NCLOSED [ BY ]  [X] 'char' ]

Замечание: переменная char может быть заключена либо в простые, либо в двойные кавычки. Если используется X'char', то char рассматривается как символ ASCII в 16-ом формате. Ключевое слово TERMINATED (ограничен) должно следовать перед ключевым словом NCLOSED (заключен).

**Указание файла данных**

Для полного определения файла данных нужно указать имя файла и (необязательно) режим чтения файла. Вы можете определить несколько файлов, используя несколько операторов INFILE.

**Имя файла**

Для указания файла, содержащего загружаемые данные, используется ключевое слово INFILE или INDDN, за которым следует имя файла и (необязательно) режим чтения файла. Имя файла, указанное в INFILE или INDDN, отвергает имя файла, указанное в командной строке. Если имя файла не указано, по умолчанию принимается имя управляющего файла с расширением имени или типом файла DAT. Если загружаемые данные находятся в управляющем файле, то вместо имени файла нужно указать "\*".

{ INFILE | INDDN }  { filename | \* }

Любое ключевое слово можно использовать INFILE или INDDN.

FILENAME

Имя файла, содержащего загружаемые данные. Можно использовать любое имя файла, допустимое в OC. Оно может быть заключено в двойные кавычки. Если имя содержит пробел или знаки пунктуации, кавычки обязательны.

**Указание нескольких файлов данных**

Для загрузки данных из нескольких файлов данных за один прогон SQL\*Loader нужно использовать оператор INFILE для каждого файла данных.

Ниже приведены примеры определения файлов данных:

 INFILE \*   указывает, что данные содержатся в самом управляющем файле.

 INFILE WHIRL  указывает, что данные содержатся в файле с именем WHIRL и подразумеваемое расширение или тип файла DAT.

INFILE путь\_к\_файлу\_WHIRL.csv   указывает, что данные содержатся в файле с именем WHIRL и подразумеваемое расширение .csv.

**Перезапись или добавление записей в существующую таблицу**

У вас есть 3 варианта загрузки данных при выполнении SQL\*Loader:

 – Загрузить ваши данные несмотря на то, что таблица уже содержит данные.

 – Заменить любые существующие данные вашими.

 – Загрузить данные, если только таблица пуста. Если в это случае таблица содержит данные, то возникает ошибка и загрузка прекращается (Эта возможность используется по умолчанию).

Итак для каждой загружаемой таблицы вы можете указать одну из следующих опций:

[ APPEND | REPLACE | INSERT ]

**APPEND** Если в таблице уже существуют данные, то SQL\*Loader добавляет новые строки в нее. Если в таблице данных нет, новые строки просто загружаются.

**INSERT** Значение по умолчанию. Требует, чтобы таблица была пуста перед загрузкой. Если таблица содержит строки, то SQL\*Loader возвращает ошибку и прекращает выполнение.

**REPLACE** Если в таблице уже существуют данные, то SQL\*Loader заменяет их новыми строками. Если в таблице данных нет, новые строки просто загружаются. Таким образом, старые строки теряются. Этот аргумент требует, чтобы пользователь, вызывающий SQL\*Loader, имел полномочия DELETE для таблицы. Вы не можете восстановить данные, которые содержались в таблице до загрузки, если только они не были сохранены утилитой EXPORT или каким-нибудь другим способом.

Вы можете использовать одну опцию для всех загружаемых таблиц, помещая ее перед всеми предложениями INTO TABLE. Либо вы можете указать опцию для отдельной таблицы, помещая ее после требуемого предложения INTO TABLE. В первом случае опция действует для всех таблиц, не имеющих своих собственных опций.

**Определение таблиц и колонок**

Раздел оператора LOAD DATA, в котором определяются таблицы, колонки и типы данных, называется блоком INTO TABLE. Чтобы загрузить несколько таблиц, вы должны включить по одному блоку INTO TABLE для каждой загружаемой таблицы.

**Указание имен таблиц**

Блок INTO TABLE начинается ключевыми словами INTO TABLE, за которыми следует имя таблицы ORACLE, принимающей данные.

INTO TABLE tablename  [ INSERT | APPEND | REPLACE ]

Таблица должна уже существовать. Имя таблицы должно быть заключено в двойные кавычки, если оно совпадает с ключевым словом SQL или SQL\*Loader, если оно содержит специальные символы или если просто есть желание.

INTO TABLE SCOTT."COMMENT"

INTO TABLE SCOTT."comment"

INTO TABLE SCOTT."-COMMENT"

Имя пользователя, выполняющего SQL\*Loader, должно иметь полномочия INSERT для таблицы, а если пользователь не является владельцем таблицы, имя владельца должно указываться перед именем таблицы:

INTO TABLE HR.EMP

Блок INTO TABLE может содержать ряд опций для загрузки таблицы. Например, вы можете указать различные опции (INSERT, APPEND, REPLACE) для каждой таблицы в зависимости от того, что SQL\*Loader должен делать, если в таблице уже имеются данные.

**Выбор записей для загрузки**

Вы можете выбрать: или загрузить, или пропустить логическую запись - используя предложение WHEN для проверки условия в записи. Предложение WHEN указывается после имени таблицы и содержит одно или несколько условий.

WHEN (field\_condition) [ AND (field\_condition) ].

FIELD\_CONDITION

Условие для поля. Это условие аналогично условию в предложении CONTINUEIF, но имеет 2 важных отличия.

Во-первых, позиции ссылаются на логическую запись, а не на физическую.

Во-вторых, вы можете указывать либо позиции в логической записи, либо имена колонок в загружаемой таблице. После того, как значение колонки определено, проверяется условие WHEN. Строка будет вставлена в таблицу только в том случае, если условие WHEN выполняется. В следующем примере указывается, что каждая запись со значением "q" в 5-й колонке должно быть загружена:

WHEN (5) = 'q'

Предложение WHEN может содержать несколько условий, соединенных связкой AND. Скобки необязательны, но могут использоваться для ясности в случае нескольких условий. Например:

WHEN ( DEPTNO = '10' ) AND ( JOB = 'SALES' )

**Формирование уникального последовательного номера**

Функция SEQUENCE может быть использована, чтобы обеспечить уникальное значение для любой отдельной колонки. SEQUENCE возрастает для каждой записи, которая загружается или отбрасывается. Она не возрастает для пропускаемых записей. Она имеет 2 аргумента. Первый - начальное значение, второй - шаг возрастания. Если начальное значение есть целое положительное n, то первая запись, добавляемая в таблицу, будет иметь значение n для этой колонки; значение для следующей строки будет увеличено на шаг.

Комбинация "имя колонки - функция SEQUENCE" представляет полное описание колонки:

              column\_name SEQUENCE  ( { n | MAX | COUNT }  [ , increment ] )

**SEQUENCE**

Функция SEQUENCE используется для определения значения колонки.

**N**

Указывает начальное значение n.

**COUNT**

В качестве начального значения принимается число строк, уже имеющихся в таблице, плюс шаг.

**MAX**

В качестве начального значения принимается максимальное значение колонки в существующих таблицах плюс шаг.

**INCREMENT**

Шаг возрастания. По умолчанию равен 1. Если строка отброшена (например, из-за ошибки формата или ошибки ORACLE), то значение, генерируемое функцией SEQUENCE, не корректируется. Так, если колонка в 4 последовательных строках должна получить значения 10, 12, 14 и 16, а вторая запись отбрасывается, то 3 строки, вставленные в таблицу, будут иметь значения 10, 14 и 16, а не 10, 12 и 14. Это позволяет сохранить последовательность вставок, несмотря на ошибки данных. Если вы отметите отброшенные записи и повторно загрузите их, вы можете вручную установить значения колонок в соответствии с требуемой последовательностью.

**Указание позиции поля данных**

Чтобы загрузить данные из файла данных, SQL\*Loader должен знать, где размещено поле и какова его длина. Чтобы указать позицию поля в логической записи, используйте предложение POSITION в определении колонки. Позиция может быть указана явно или относительно предыдущего поля. Аргументы POSITION должно быть заключены в круглые

скобки.

POSITION ( { start [ :end ] | \* | [ +n ] } )

**START**

Указывает начальную позицию поля данных в логической записи. Позиции логической записи нумеруются с 1.

**END**

Указывает конечную позицию поля данных в логической записи. Допускается использовать двоеточие или дефис (start:end или start-end). Если вы опуститеnd, длина поля определяется по типу данных в файле данных (см. разделы для каждого типа данных; заметим, что если данные CHAR указаны без start иnd, то длина поля считается равной 1). Если невозможно определить длину поля по типу данных, выдается сообщение об ошибке.

**\***

Указывает, что поле данных следует непосредственно за предыдущим полем.

Если \* используется для первого поля данных, то считается, что оно начинается с начала логической записи. Если \* используется без start и end, длина поля определяется по типу данных.

**E+n**

Смещение, указанное как +n, может быть использовано с \*, чтобы определить положение поля относительно предыдущего. n символов будет пропущено перед чтением значения для этого поля. Предложение POSITION может быть полностью опущено. В этом случае для спецификации поля используется значение POSITION(\*). Если при загрузке нескольких таблиц вы указываете POSITION(\*) для первой колонки первой таблицы, то позиция определяется с начала логической записи. Если вы указываете POSITION(\*) для первой колонки следующих таблиц, позиция определяется относительно последней колонки предыдущей таблицы. Поэтому позиции во втором и следующих блоках INTO TABLE не установлены с начала логической записи, если вы не потребуете этого явно.

Выход простой:

вместо POSITION(\*) или POSITION(\*+n) для первого поля следующего блока INTO TABLE используйте POSITION(1) или POSITION(n). Это позволяет загружать несколько таблиц, когда данные для каждой таблицы находятся в одной и той же логической записи. Для каждой следующей таблицы можно либо продолжить обработку логической записи, либо начать сначала.

**BEGINDATA**

Если вы опустите это ключевое слово, SQL\*Loader попытается интерпретировать ваши данные как управляющую информацию, и вы получите сообщение об ошибке. Если данные находятся в отдельном файле, достижение конца управляющего файла указывает, что управляющая информация закончилась, и ключевое слово BEGINDATA не нужно.

Данные могут быть фиксированной длины или с разделителями.

### Формат с фиксированной длиной записей

Если все поля записей имеют идентичную длину, то надо указать границы полей в управляющем файле загрузчика. Например:

LOAD DATA

INFILE 'E:\database\load\books1.dat'

INTO TABLE "LIBRARY".books1 (

Id POSITION (01:04) INTEGER EXTERNAL,

ISBN POSITION (05:18) CHAR,

OWNER\_ID POSITION (22:24) INTEGER EXTERNAL,

PUBLISHER\_ID POSITION (25:29) INTEGER EXTERNAL,

REM POSITION (30:40) CHAR)

В этом примере указаны: директива загрузки, местонахождение входного файла, описание таблицы и описание формата данных. Этой информации вполне достаточно для успешной загрузки данных.

Типы данных (INTEGERXTERNAL, CHAR, DECIMALXTERNAL) указывают тип данных поля в файле, а не соответствующей колонки в таблице

**Формат с переменной длиной записей**

Если колонки в файле данных разного размера, то необходимо указать разделитель колонок. Этот разделитель является символом (или набором символов), который указывает SQL\*Loader-у где колонки начинаются, и где они заканчиваются. Далее пример управляющего файла с записями переменной длины:

LOAD DATA

INFILE 'D:\database\load\books2.dat'

APPEND

INTO TABLE "LIBRARY".books2

FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'

(id, ISBN CHAR, owner\_id, publisher\_id, rem CHAR)

**Формат с переменной длиной записей с данными**

Пример:

LOAD DATA

   INFILE \*

   INTO TABLE DEPT

   FIELDS TERMINATED BY ', ' OPTIONALLYNCLOSED BY '"'

   (DEPTNO, DNAME, LOC)

   BEGINDATA

      12, RESEARCH, "SARATOGA"

      10, "ACCOUNTING", CLEVELAND

      11, "ART", SALEM

      13, FINANCE, "BOSTON"

      21, "SALES", PHILA.

      22, "SALES", ROCHESTER

      42, "INT'L", "SAN FRAN"