SQL

Базовый уровень

Основные понятия и объекты СУБД

Типы данных (Data types)

Все данные хранятся в определенном формате, который называется типом данных (data type). Типы данных могут быть классифицированы по таким основным категориям:

Тип данных	Объявления
Символьный	CHAR | VARCHAR
Битовый	BIT | BIT VARYING
Точные числа	NUMERIC | DECIMAL | INTEGER | SMALLINT
Округленные числа	FLOAT | REAL | DOUBLE PRECISION
Дата/время	DATE | TIME | TIMESTAMP
Интервал	INTERVAL

Типы данных (Data types)

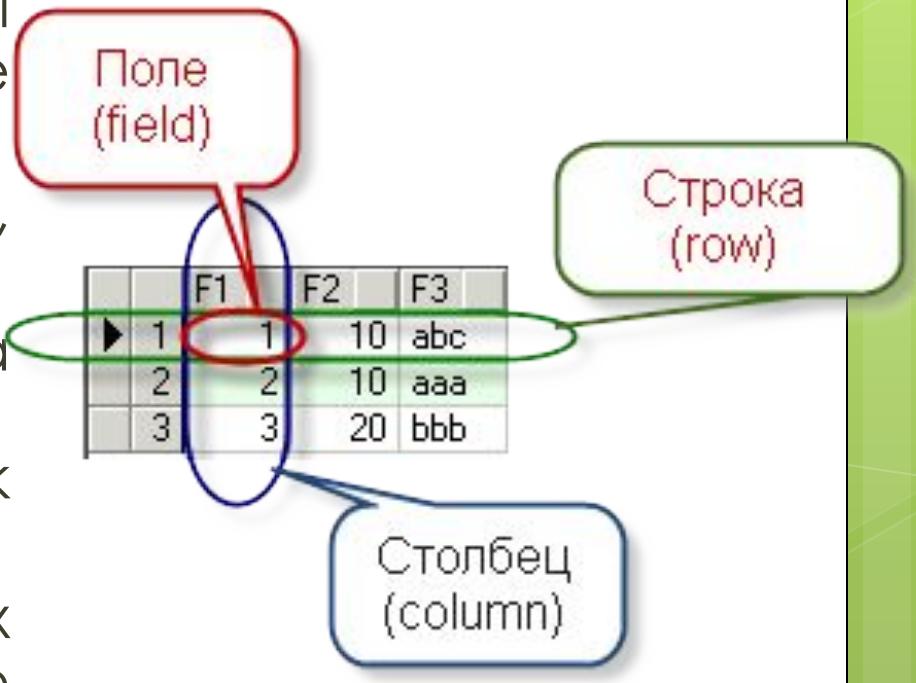
Отдельно выделяют большие бинарные объекты (*binary large object – BLOB*), которые могут хранить данные неограниченного размера. Тип *BLOB* это расширение стандартной реляционной модели, которая обычно обеспечивает только типы данных фиксированной длины. Так как BLOB столбцы часто содержат большие и переменные объемы данных, BLOB столбцы хранятся в отдельных сегментах.

Таблицы (Tables)

Реляционные базы данных хранят все данные в таблицах.

Таблица это структура, состоящая из:

- множества неупорядоченных горизонтальных строк (*rows*),
- каждая из которых содержит одинаковое количество вертикальных столбцов (*colums*).



Генераторы (Generators)

Генератор – это механизм, который создает последовательный уникальный номер, который автоматически вставляется в столбец базой данных, когда выполняются операции INSERT или UPDATE. Генератор обычно применяется для создания уникальных значений, вставляемых в столбец, который используется как PRIMARY KEY.

Первичные и внешние ключи (primary key, foreign key)

Ключ – это столбец (несколько столбцов), добавляемый к таблице и позволяющий установить связь с записями в другой таблице. Существуют ключи двух типов:

- первичные
- вторичные (внешние).

Первичные и внешние ключи (primary key, foreign key)

Первичный ключ – это одно или несколько полей (столбцов), комбинация значений которых однозначно определяет каждую запись в таблице. Первичный ключ

- не допускает значений Null ;
- всегда должен быть уникальным;
- никогда не меняется.

Первичный ключ используется для связывания таблицы с внешними ключами в других таблицах.

Первичные и внешние ключи (primary key, foreign key)

Внешний (вторичный) ключ – это одно или несколько полей (столбцов) в таблице, содержащих ссылку на поле или поля первичного ключа в другой таблице.

Внешний ключ определяет способ объединения таблиц.

Из двух логически связанных таблиц одну называют таблицей первичного ключа или *главной(родительской)* таблицей, а другую таблицей вторичного (внешнего) ключа или *подчиненной* (дочерней) таблицей.

Первичные и внешние ключи (primary key, foreign key)

Используется три типа первичных ключей:

- *Поле счетчика* (Тип данных «Генератор»). Тип данных поля в БД, в котором для каждой добавляемой в таблицу записи в поле автоматически заносится уникальное числовое значение.

Первичные и внешние ключи (primary key, foreign key)

- *Простой ключ*. Если поле содержит уникальные значения, такие как коды или инвентарные номера, то это поле можно определить как первичный ключ.

Первичные и внешние ключи (primary key, foreign key)

- *Составной ключ*. В случаях, когда невозможно гарантировать уникальность значений каждого поля, существует возможность создать ключ, состоящий из нескольких полей. Чаще всего такая ситуация возникает для таблицы, используемой для связывания двух таблиц многие - ко - многим.

Первичные и внешние ключи (primary key, foreign key)

Внешний ключ может быть создан только после создания соответствующего первичного ключа. Внешние ключи имеют следующие свойства:

- внешний ключ должен содержать такое же число колонок, такого же типа и в том же порядке следования, что и соответствующий первичный ключ;
- имена колонок внешнего ключа и их значения по умолчанию могут отличаться от используемых в соответствующем первичном ключе (в том числе иметь null-значения);

Первичные и внешние ключи (primary key, foreign key)

- таблица может иметь любое число внешних ключей;
- упорядочение значений колонок внешнего ключа в его индексе может отличаться от соответствующего первичного ключа;

Поддержка ссылочной целостности посредством внешних ключей не требует соответствующего индекса для внешнего ключа.

Индексы (Indexes)

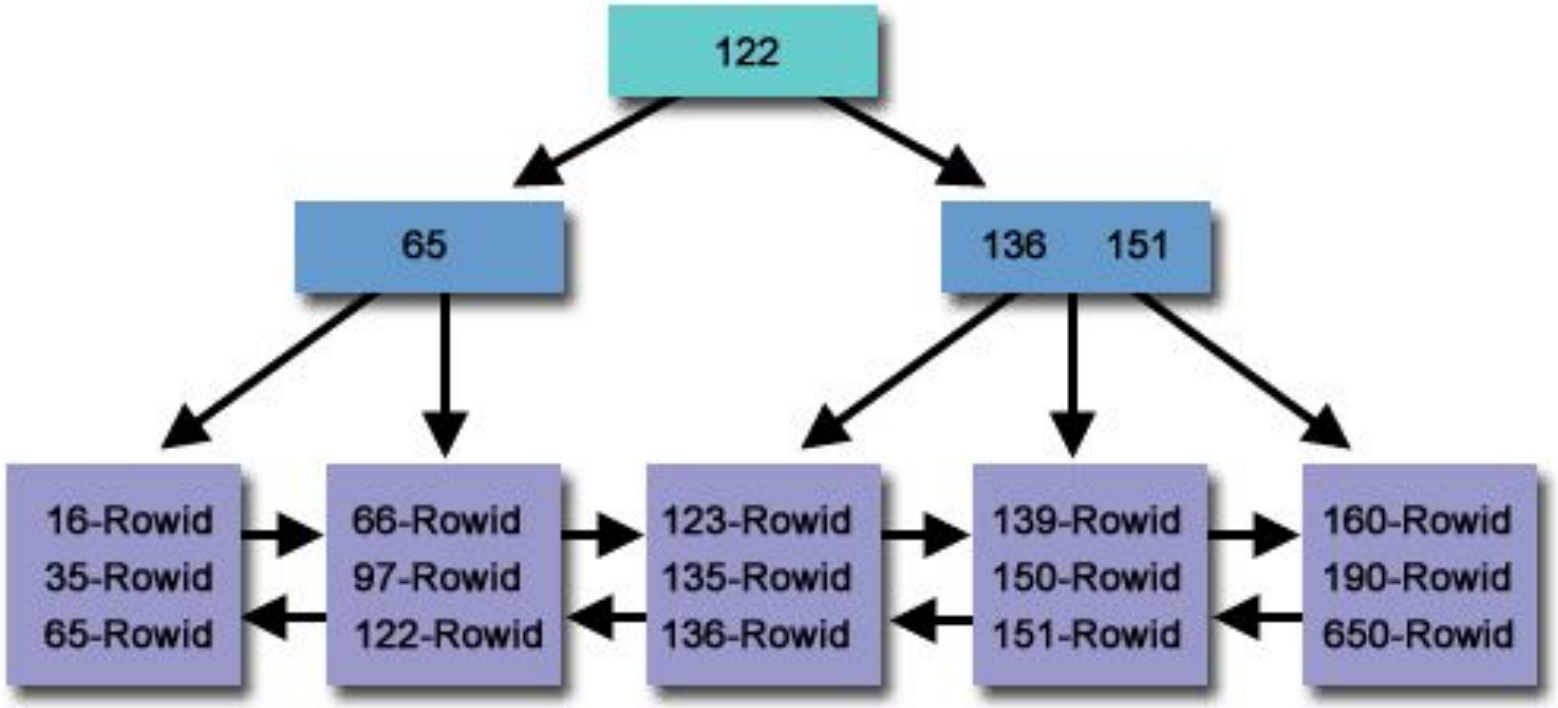
Индексы это механизм для улучшения быстродействия поиска данных. Индекс определяет столбцы, которые могут быть использованы для эффективного поиска и сортировки в таблице.

Уникальный индекс для первичного ключа отношения называется *первичным* индексом.

Индексы (Indexes)

Семейство *B-Tree индексов* – это наиболее часто используемый тип индексов, организованных как сбалансированное дерево, упорядоченных ключей. Они поддерживаются практически всеми СУБД как реляционными, так нереляционными, и практически для всех типов данных.

Индексы (Indexes)



Индексы (Indexes). Битовые индексы

В индексе на основе битовых карт запись использует битовую карту для ссылки на большое количество строк одновременно. Такие индексы подходят для данных с небольшим количеством различных значений, которые обычно только читаются.

Значение/Строка	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
ANALYST	0	0	0	0	0	0	0	1	0	1	0	0	1	0
CLERK	1	0	0	0	0	0	0	0	0	0	1	1	0	1
MANAGER	0	0	0	1	0	1	1	0	0	0	0	0	0	0
PRESIDENT	0	0	0	0	0	0	0	0	1	0	0	0	0	0
SALESMAN	0	1	1	0	1	0	0	0	0	0	0	0	0	0

Индексы (Indexes). Функциональные индексы

Эти индексы на основе B*-дерева или битовых карт хранят вычисленный результат применения функции к столбцу или столбцам строки, а не сами данные строки. Это можно использовать для ускорения выполнения запросов вида:

```
SELECT *
FROM T
WHERE ФУНКЦИЯ(СТОЛБЕЦ) = ЗНАЧЕНИЕ,
```

Ограничения (Constraints)

Существуют три основных типа ограничений, используемых в реляционной БД:

- ограничения целостности данных (data integrity constraints) – относятся к значениям данных в некоторых колонках и определяются в спецификации колонки с помощью элементов SQL NOT NULL, UNIQUE, CHECK;
- ограничения целостности ссылок (referential constraints) – относятся к связям между таблицами на основе связи первичного и внешнего ключей;
- ограничения первичного ключа – относятся к значениям данных в колонках первичного ключа таблицы.

Ограничения (Constraints)

Ограничение	Описание
CHECK	гарантирует, что значения находятся в границах специфицированного интервала, задаваемого предикатом
DEFAULT	помещает значение по умолчанию в колонку. Гарантирует, что колонка всегда имеет значение
FOREIN KEY	гарантирует, что значения существует как значение в колонке первичного ключа другой таблицы. Обеспечивает процедуры удаления дочерних строк при удалении связанных с ней родительских
NOT NULL	гарантирует, что колонка всегда содержит значение
PRIMARY KEY	гарантирует, что колонка всегда содержит значение и оно уникально в таблице
UNIQUE	гарантирует, что значение будет уникальным в таблице

Представления (view)

Представление – это виртуальная таблица, которая не сохранена физически в БД, но ведет себя точно также как "реальная" таблица. Может содержать данные из одной или более таблиц или других представлений и используется для хранения часто используемых запросов или множества запросов в БД.

Сохраненные процедуры (Stored procedures)

Сохраненные процедуры – это отдельные программы, написанные на языке процедур и триггеров, который является расширением SQL.

Сохраненные процедуры могут получать входные параметры, возвращать значения приложению и могут быть вызваны явно из приложения или подстановкой вместо имени таблицы в инструкции SELECT.

Сохраненные процедуры (Stored procedures)

Сохраненные процедуры обеспечивают следующие возможности:

- модульный проект: могут быть общими для приложений, которые обращаются к той же самой БД, что позволяет избегать повторяющегося кода, и уменьшает размер приложений.
- упрощают сопровождение приложений: при обновлении процедур, изменения автоматически отражаются во всех приложениях, которые используют их без необходимости их повторной компиляции и сборки.
- улучшают эффективность работы (выполняются сервером, что снижает сетевой трафик).

Триггеры (Triggers)

Триггеры это отдельная программа, ассоциированная с таблицей или представлением, которая автоматически выполняет действия, при

- добавлении,
- изменении,
- удалений строки

в таблице или представлении.

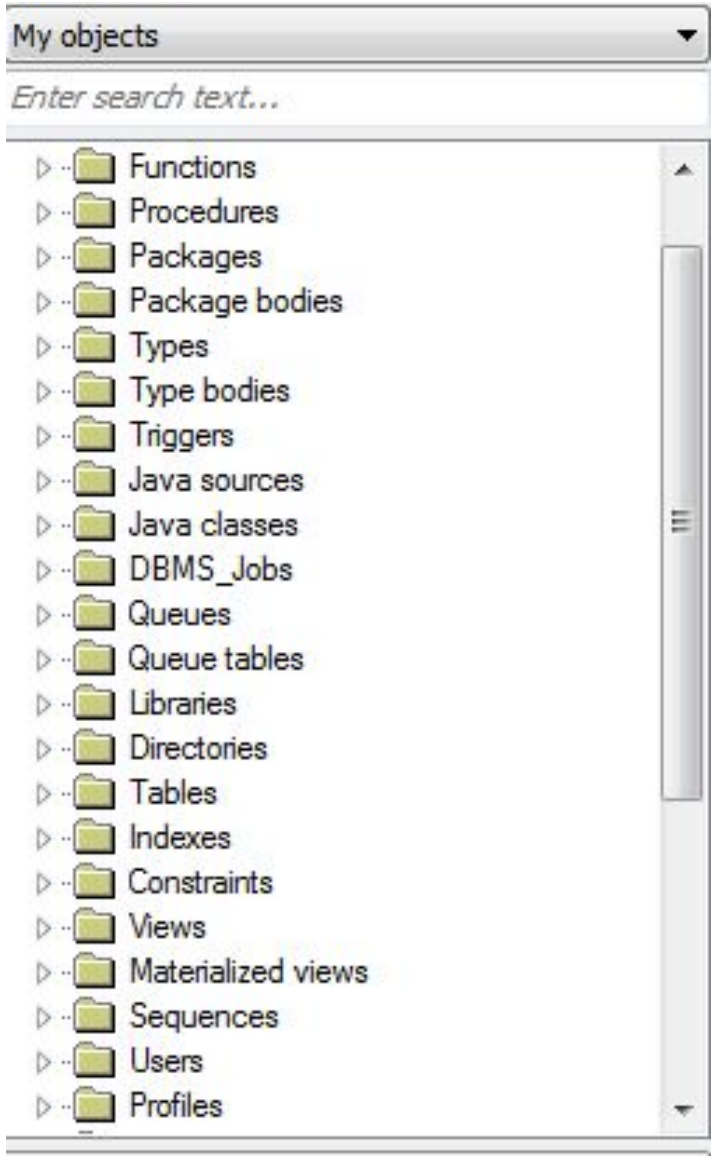
Триггеры (Triggers)

Триггеры могут обеспечивать следующие возможности:

- автоматическое ограничение ввода данных, чтобы гарантировать, что пользователь ввел только допустимые значения в поля столбцов;
- упрощение сопровождения приложений, так как изменение в триггере автоматически отражается во всех приложения, которые используют таблицы со связанными с ними триггерами;
- автоматическое документирование изменений таблицы – приложение может управлять логом изменений с помощью триггеров, которые выполняются всякий раз, когда происходит изменение таблицы.

PL/SQL Developer

PL/SQL Developer - Интегрированная среда разработки, которая была специально написана для разработки программных объектов для баз данных Oracle.



Основы SQL

Описание основных операторов SQL

SQL(Structured query language) состоит из набора команд манипулирования данными в реляционной БД, которые позволяют создавать объекты реляционной базы данных, модифицировать данные в таблицах (вставлять, удалять, исправлять), изменять схемы отношений базы данных, выполнять вычисления над данными, делать выборки из базы данных, поддерживать безопасность и целостность данных.

Описание основных операторов SQL

Весь набор команд SQL можно разбить на следующие группы:

- команды определения данных (DDL – Data Defininion Language);
- команды манипулирования данными (DML – Data Manipulation Language);
- команды выборки данных (DQL– Data Query Language);
- команды управления транзакциями;
- команды управления данными.

Описание основных операторов SQL

Команды определения данных объектов	
ALTER TABLE	Изменяет описание таблицы (схему отношения)
CREATE INDEX	Создает индекс для таблицы
CREATE SEQUENCE	Создает последовательность
CREATE TABLE	Определяет таблицу
CREATE TRIGGER	Создает триггер в БД
CREATE VIEW	Определяет представление на таблицах
DROP INDEX	Физически удаляет индекс из баз данных
DROP SEQUENCE	Удаляет последовательность
DROP TABLE	Физически удаляет таблицу из базы данных
DROP VIEW	Удаляет представление

Описание основных операторов SQL

Команды манипулирования данными	
DELETE	Удаляет одну или более строк из таблицы базы данных
INSERT	Вставляет одну или более строк в таблицу базы данных
UPDATE	Обновляет значения колонок в таблице базы данных
Команды выборки данных	
SELECT	Выполняет запрос на выборку данных из таблиц и представлений
UNION	Объединяет в одной выборке результаты выполнения двух или более команд SELECT

Описание основных операторов SQL

Команды управления транзакциями	
COMMIT	Завершает транзакцию и физически актуализирует состояние базы данных
ROLLBACK	Завершает транзакцию и возвращает текущее состояние базы данных на момент последней завершенной транзакции и контрольной точки
SAVEPOINT	Назначает контрольную точку внутри транзакции

Создание таблицы

CREATE TABLE имя_таблицы
{(*имя_столбца тип_данных* [NOT NULL][UNIQUE] [DEFAULT <значение>]
[CHECK (<условие_выбора>)][,...n]}
[CONSTRAINT имя_ограничения]
[PRIMARY KEY (имя_столбца [,...n])
{[UNIQUE (имя_столбца [,...n])}
[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n])
REFERENCES имя_род_таблицы [(столбец_род_табл [,...n])],
[MATCH {PARTIAL | FULL}]
[ON UPDATE {CASCADE| SET NULL |SET DEFAULT
|NO ACTION}]
[ON DELETE {CASCADE| SET NULL |SET DEFAULT
|NO ACTION}]
{[CHECK(<условие_выбора>)][,...n]})

Создание таблицы

- Обязательные данные

Для некоторых столбцов требуется наличие в каждой строке таблицы конкретного и допустимого значения, отличного от опущенного значения или значения NULL. Для заданий ограничений подобного типа стандарт SQL предусматривает использование спецификации NOT NULL.

{(*имя_столбца тип_данных*
[NOT NULL][UNIQUE] [DEFAULT <значение>]
[CHECK (<условие_выбора>)][,...n]}

id_book INTEGER not null

Создание таблицы

- Требования конкретного предприятия

Обновления данных в таблицах могут быть ограничены существующими в организации требованиями (бизнес-правилами). Стандарт SQL позволяет реализовать бизнес-правила предприятий с помощью предложения CHECK и ключевого слова UNIQUE.

```
constraint UK_CODE_READER unique (CODE_READER);

constraint CKC_IS_READABLE_BOOK_EXA
                check (IS_READABLE in (0,1))
```

Создание таблицы

- Ограничения для доменов полей

Стандарт SQL предусматривает два различных механизма определения доменов. Первый состоит в использовании предложения CHECK, позволяющего задать требуемые ограничения для столбца или таблицы в целом, а второй предполагает применение оператора CREATE DOMAIN.

CREATE DOMAIN RoomType AS CHAR(1)
CHECK(VALUE IN (□S□, □F□, □D□));

Создание таблицы

- Целостность сущностей

Первичный ключ таблицы должен иметь уникальное непустое значение в каждой строке. Стандарт SQL позволяет задавать подобные требования поддержки целостности данных с помощью фразы PRIMARY KEY.

constraint PK_BOOK_EXAMPLE primary key (ID_BOOK_EXAMPLE)

Создание таблицы

- Ссылочная целостность

Стандарт SQL предусматривает механизм определения внешних ключей с помощью предложения FOREIGN KEY, а фраза REFERENCES определяет имя родительской таблицы, т.е. таблицы, где находится соответствующий потенциальный ключ.

При использовании этого предложения система отклонит выполнение любых операторов INSERT или UPDATE, с помощью которых будет предпринята попытка создать в дочерней таблице значение внешнего ключа, не соответствующее одному из уже существующих значений потенциального ключа родительской таблицы.

Создание таблицы

- Ссылочная целостность

Если пользователь предпринимает попытку удалить из родительской таблицы строку, на которую ссылается одна или более строк дочерней таблицы, язык SQL предоставляет следующие возможности:

- CASCADE
- SET NULL
- SET DEFAULT
- NO ACTION (по умолчанию)

Создание таблицы

- Ссылочная целостность

```
constraint FK_BOOK_EXAMPLE__D_BOOK
   foreign key (ID_BOOK)
   references D_BOOK (ID_BOOK);

constraint FK_BOOK_AUTHOR__D_AUTHOR
foreign key (ID_AUTHOR)
  references D_AUTHOR (ID_AUTHOR) on delete cascade;

constraint FK_BOOK_AUTHOR__D_BOOK
 foreign key (ID_BOOK)
  references D_BOOK (ID_BOOK) on delete set null;
```

Создание таблицы

- Значение по умолчанию

Необязательная фраза DEFAULT предназначена для задания принимаемого по умолчанию значения, когда в операторе INSERT значение в данном столбце будет отсутствовать.

date_start DATE default SYSDATE not null,

Создание таблицы

```
CREATE TABLE MANAGEMENT (
    MANAGNO  INT      NOT NULL,
    EMPNO    INT,
    JOB          INT,
    PRIMARY KEY (MANAGNO),
    FOREIGN KEY fnkey (EMPNO)
    REFERENCES EMPLOYEE ON DELETE CASCADE);

CREATE UNIQUE INDEX ndxmng ON
MANAGEMENT(MANAGNO);
```

Изменение и удаление таблицы

Для внесения изменений в уже созданные таблицы стандартом SQL предусмотрен оператор ALTER TABLE, предназначенный для выполнения следующих действий:

- добавление в таблицу нового столбца;
- удаление столбца из таблицы;
- добавление в определение таблицы нового ограничения;
- удаление из определения таблицы существующего ограничения;
- задание для столбца значения по умолчанию;
- отмена для столбца значения по умолчанию.

Изменение и удаление таблицы

```
ALTER TABLE имя_таблицы
[ADD [COLUMN]имя_столбца тип_данных  [ NOT NULL ][UNIQUE]
              [DEFAULT <значение>][ CHECK (<условие_выбора>)]]
[DROP [COLUMN] имя_столбца [RESTRICT | CASCADE ]]
[ADD [CONSTRAINT [имя_ограничения]]
        [{PRIMARY KEY (имя_столбца [,...n])
        | [UNIQUE (имя_столбца [,...n])}
| [FOREIGN KEY (имя_столбца_внешнего_ключа [,...n])
   REFERENCES род_таблицы    [(поле_род_таблицы [,...n])],
[ MATCH {PARTIAL | FULL}
   [ON UPDATE {CASCADE| SET NULL | SET DEFAULT | NO ACTION}]
   [ON DELETE {CASCADE| SET NULL |   SET DEFAULT | NO ACTION}]
   | [CHECK(<условие_выбора>)][,...n]}]
[DROP CONSTRAINT имя_ограничени     [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT <значение>]
[ALTER [COLUMN] DROP DEFAULT]
```

Изменение и удаление таблицы

```
--добавление первичного ключа
ALTER TABLE READER_BOOK
  ADD CONSTRAINT PK_READER_BOOK
        PRIMARY KEY (ID_READER_BOOK);

-- добавление ссылочного ключа
ALTER TABLE READER_BOOK
    ADD CONSTRAINT FK_TABLE
    FOREIGN KEY (ID_READER)
    REFERENCES D_READER (ID_READER);
```

Создание индекса

CREATE [UNIQUE] [CLUSTERED HASHED] INDEX имя_индекса

ON имя_таблицы (имя_колонки [ASC | DESC] [, имя_колонки s])

create unique index IDX_CODE_READER on D_READER (CODE_READER)

Простые запросы

Простые запросы

Оператор SELECT – один из наиболее важных и используемых операторов SQL. Он позволяет производить выборки данных из БД и преобразовывать к нужному виду полученные результаты.

Оператор SELECT – полностью абстрагирован от вопросов представления данных, всё внимание при его применении сконцентрировано на проблемах доступа к данным.

Простейший вид оператора

В простейшем случае оператор SELECT имеет вид:

SELECT {* | <поле1> [, <поле2> …]}
FROM <таблица1>[, <таблица2> …]

Простейший вид оператора

```
select *
from d_book;
```

D_BOOK		
ID_BOOK	INTEGER	<pk>
NAME_BOOK	VARCHAR2(250)	
DESCRIPTION	VARCHAR2(4000)	
PAGE_NUM	INTEGER	
BOOK_YEAR	VARCHAR2(4)	

```
select id_book, name_book,
       description, page_num,
       book_year
from d_book;
```

Простейший вид оператора

В списке могут использоваться не только поля, но и любые выражения от них с арифметическими операциями +, -, *,/. После выражения может записываться псевдоним выражения в форме **AS <псевдоним>**.

В качестве псевдонима может использоваться любой идентификатор. Таким простым образом создаются аналоги вычисляемых полей.

Использование псевдонимов таблиц

В запросе SELECT можно объединить данные нескольких таблиц.

Каждое имя поля должно предваряться ***ссылкой*** на таблицу, к которой она относится. В операторах, работающих с несколькими таблицами, обычно каждой таблице даётся ***псевдоним***, сокращающий ссылки на таблицы.

При помощи псевдонимов возможно самообъединение таблиц.

Простейший вид оператора

```
select rb.date_start + rb.term as date_end_pl,
       (rb.date_start + rb.term)
from reader_book rb
```

	DATE_END_PL	(RB.DATE_START+RB.TERM)
1	5/26/2001 12:28:29 PM	5/26/2001 12:28:29 PM
2	9/16/1993 4:00:44 AM	9/16/1993 4:00:44 AM
3	6/18/1993 3:47:09 AM	6/18/1993 3:47:09 AM
4	7/14/1996 7:08:52 AM	7/14/1996 7:08:52 AM

Определение сортировки ORDER BY

Набор данных, выдаваемый в результате выполнения оператора SELECT, в общем случае возвращается в неотсортированном виде. Определить, по каким полям необходимо сортировать записи в результирующем наборе данных, можно при помощи оператора ORDER BY, имеющего следующий формат:

ORDER BY <список столбцов>[asc | desc]

Использование WHERE

С использованием оператора WHERE оператор SELECT имеет следующий формат:

SELECT {* | <поле1> [, <поле2> …]}
FROM <таблица1>[, <таблица2> …]
WHERE <условия поиска>

Использование WHERE

В набор данных, который возвращается как результат выполнения оператора SELECT, будут включаться только те записи, которые удовлетворяют условию WHERE. Условие может включать:

- имена полей (кроме вычисляемых),
- константы,
- логические выражения с арифметическими операциями,
- логические операции **and**, **or**, **not**,

Использование WHERE

- операции отношения:

=	равно
>	больше
>=, !<	больше или равно
<	меньше
<=, !>	меньше или равно
!=, <>	не равно
Like	наличие заданной последовательности символов
between ... and	диапазон значений
in	соответствие элементу множества
is null	сравнение с неопределённым значением

Пример

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY > 40000 AND POSITION = 'Staff';

SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY < 40000 OR BENEFITS < 10000;

SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE POSITION = 'Manager' AND
        SALARY > 60000 OR
        BENEFITS > 12000;

SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE POSITION = 'Manager' AND
      (SALARY > 50000 OR BENEFIT > 10000);
```

Name
EMPLOYEE_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE_NUMBER
HIRE_DATE
JOB_ID
SALARY
COMMISSION_PCT
MANAGER_ID
DEPARTMENT_ID

Вывести ФИО работников, работающих в департаменте DEPARTMENT_ID =1, у которых ЗП > 2000 и зачисление на работу – не позднее 2011 года

Выбрать книги, имеющие тот же год издания и то же число страниц, что и книга с ID_BOOK = 1

D_BOOK		
ID_BOOK	INTEGER	<pk>
NAME_BOOK	VARCHAR2(250)	
DESCRIPTION	VARCHAR2(4000)	
PAGE_NUM	INTEGER	
BOOK_YEAR	VARCHAR2(4)	

Использование WHERE

Операция Like имеет следующий синтаксис:

<поле> Like '<последовательность символов>' Эта операция применима к полям типа строк и возвращает true, если в строке встретился фрагмент, заданный в операции как **'<последовательность символов>.**

Заданным символам может предшествовать или завершать их символы:

- «%», который обозначает любое количество любых символов;
- «_», который обозначает один произвольный символ.

Использование WHERE

SELECT EMPLOYEEIDNO
FROM EMPLOYEEADDRESSTABLE
WHERE LASTNAME **LIKE 'L%';**

.

Использование WHERE

Операция between ... and имеет синтаксис:

<поле> between <значение> and <значение>

и задаёт для указанного поля диапазон отбираемых значений.

Операция in имеет синтаксис:

<поле> in (<множество>)

и отбирает записи, в которых значение указанного поля является одним из элементов указанного множества.

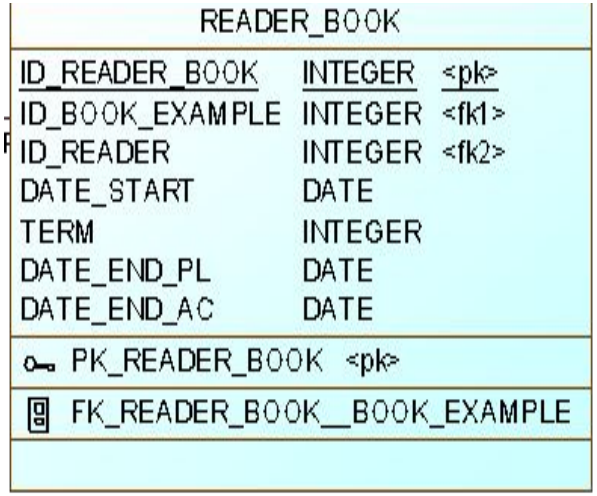
Использование WHERE

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE POSITION IN ('Manager', 'Staff');
```

```
SELECT EMPLOYEEIDNO FROM
EMPLOYEESTATISTICSTABLE
WHERE SALARY BETWEEN 30000 AND 50000;
```

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY NOT BETWEEN 30000 AND 50000;
```

Использование WHERE



Сравнение с неопределённым значением

is null

SELECT *

FROM READER_BOOK

WHERE DATE_END_AC **IS NULL**