Язык определения данных (Data Definition Language, DDL)

1. Создание, изменение и удаление таблиц
2. Ограничение значение данных
3. Поддержание ссылочной целостности

Подразделы SQL

SELECT	**Язык запросов** (Queries)
CREATE ALTER DROP	**Язык определения данных** (DDL)
INSERT UPDATE DELETE	**Язык манипулирования данными** (DML)
GRANT REVOKE	**Язык управления доступа к данным** (DCL)
COMMIT ROLLBACK	**Язык управления транзакциями** (TCL)

Пример простой схемы БД

Столбцы таблицы **Salespeople (Продавцы)**

Столбец	Описание
snum	Уникальный номер, присваиваемый каждому продавцу (номер служащего)
sname	Фамилия продавца
city	Город, где находится продавец, т. е. один из офисов компании.
comm	Комиссионное вознаграждение продавца в десятичной форме

Пример простой схемы БД

Salespeople (Продавцы)

Snum	Sname	City	Comm
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rifkin	Barcelona	.15
1003	Axelrod	New York	.10

Пример простой схемы БД

Столбцы таблицы **Customers (Покупатели)**

Столбец	Описание
cnum	Уникальный номер, присваиваемый каждому покупателю
cname	Фамилия покупателя
city	Город, где находится покупатель. Это один из офисов компании, а не место проживания покупателя
rating	Числовой код, который показывает уровень предпочтения для покупателя. NULL обозначает покупателя, которому еще не присвоен рейтинг
snum	Номер продавца (из таблицы Salespeople),

Пример простой схемы БД

Customers (Покупатели)

Cnum	Cname	City	Rating	Snum
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	NULL	1001
2008	Cisneros	San Jose	300	1007
2007	Pereira	Rome	100	1004

Пример простой схемы БД

Столбцы таблицы **Orders (Заказы)**

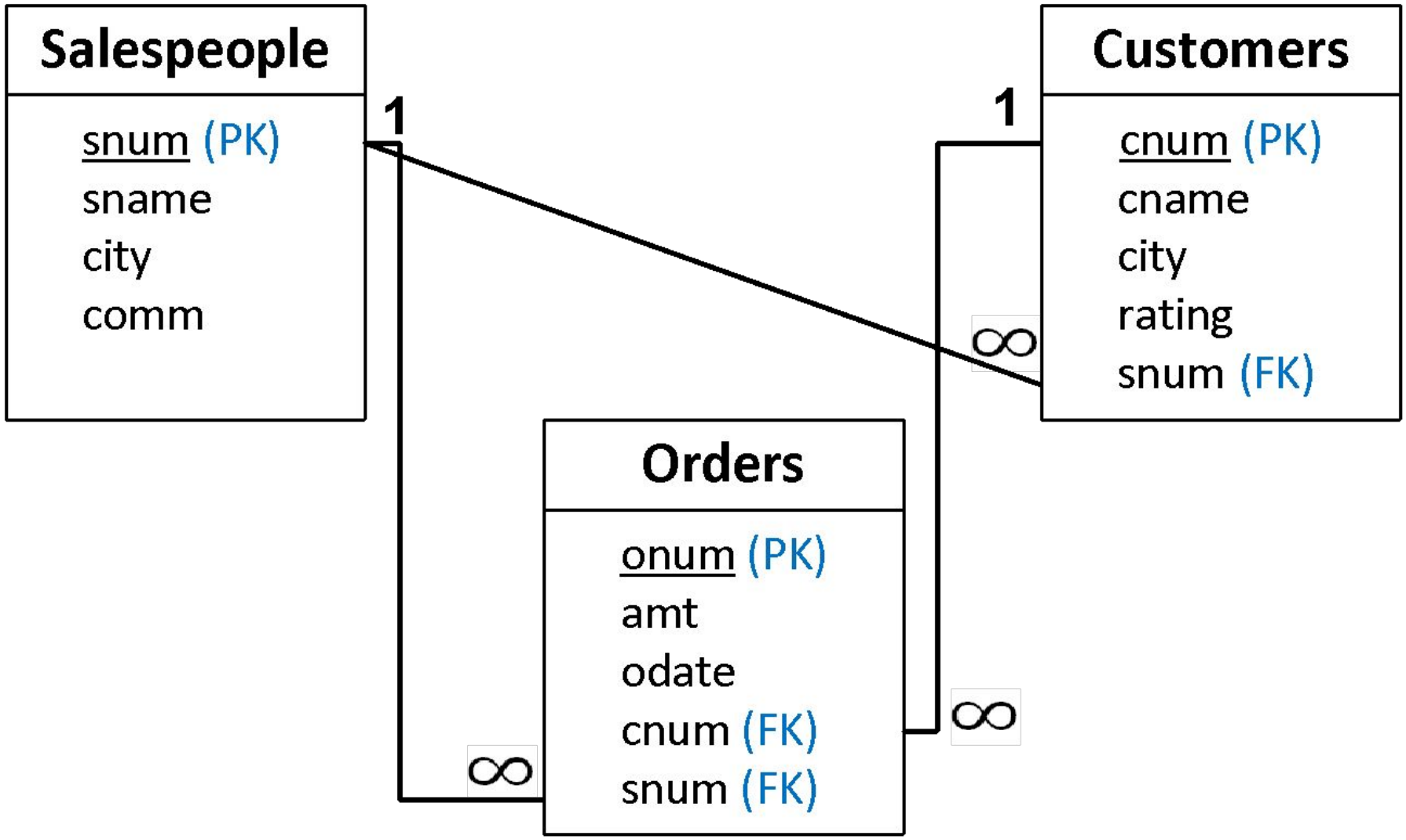
Столбец	Описание
onum	Уникальный номер, присваиваемый каждой покупке
amt	Сумма покупки
odate	Дата покупки
cnum	Номер покупателя (из таблицы Customers), делающего покупку
snum	Номер продавца (из таблицы Salespeople), совершившего продажу. Обычно это продавец, прикрепленный к покупателю в таблице Customers, но не всегда

Пример простой схемы БД

Orders (Заказы)

Onum	Amt	Odate	Cnum	Snum
3001	18.69	10/03/2000	2008	1007
3003	767.19	10/03/2000	2001	1001
3002	1900.10	10/03/2000	2007	1004
3005	5160.45	10/03/2000	2003	1002
3006	1098.16	10/03/2000	2008	1007
3009	1713.23	10/04/2000	2002	1003
3007	75.75	10/04/2000	2004	1002
3008	4723.00	10/05/2000	2006	1001
3010	1309.95	10/06/2000	2004	1002
3011	9891.88	10/06/2000	2006	1001

Пример простой схемы БД



Язык определения данных

Команды **Data Definition Language (DDL)** для работы с таблицами:

- CREATE TABLE — создание таблицы
- ALTER TABLE — изменение таблицы
- DROP TABLE — удаление таблицы

Создание таблицы

Таблицы создаются с помощью команды **CREATE TABLE**, которая:

- формирует пустую таблицу, не содержащую строк
- определяет таблицу как набор поименованных столбцов, расположенных в указанном порядке
- задает типы данных и размеры столбцов
- задает **ограничения**

Создание таблицы

Упрощенный синтаксис оператора **CREATE TABLE**

```
CREATE TABLE [схема.]имя_таблицы
({имя_столбца    тип_данных
[(размер)]},...);
```

Пример создания таблицы:

```
CREATE TABLE Salespeople
(snum   NUMBER(10),
 sname  CHAR(10),
 city   CHAR(10),
 comm   NUMBER(18,2));
```

Создание таблицы

Схема (schema) — именованная группа таблиц (а также других объектов БД)

Владелец (owner) таблицы — пользователь, который создал таблицу

Пользователи, не являющиеся владельцами таблицы, при ссылке на нее должны указать перед именем таблицы имя схемы, отделенное точкой:
<имя схемы>.<имя таблицы>

Пример: dbo.Employees

Типы данных

Тип данных	Описание
VARCHAR2(*size*)	Символьные данные переменной длины
CHAR(*size*)	Символьные данные постоянной длины
NUMBER(*p*,*s*)	Числовые данные переменной длины
DATE	Значения даты и времени
LONG	Символьные данные переменной длины до 2 гигабайтов
CLOB	Однобайтовые символьные данные до 4 гигабайтов
RAW и LONG RAW	Необработанные (“сырые“) двоичные данные
BLOB	Двоичные данные до 4 гигабайтов
BFILE	Двоичные данные, которые хранятся во внешнем файле; длина до 4 гигабайтов

Проверка создания таблицы

- Команда DESCRIBE выводит описание таблицы

```
DESCRIBE имя_таблицы
```

- Вывод таблиц, принадлежащих пользователю:

```
SELECT * FROM user_tables
```

Присвоение значений по умолчанию

Значение по умолчанию (default value, default) — это величина, которая **автоматически** вставляется в столбец таблицы в случае, если значение данного столбца не указано в операторе **INSERT**.

Использование **DEFAULT** для установки значения по умолчанию для столбца

```
CREATE TABLE  Salespeople
(snum  NUMBER(10),
 sname CHAR(10),
 city  CHAR(10) DEFAULT 'New York',
 comm  NUMBER(18,2);
```

Изменение таблицы

Оператор **ALTER TABLE** может:

- Переименовывать таблицу
- Добавлять/Изменять/Удалять **столбец/столбцы**
- Добавлять/Удалять **ограничение к таблице**
- Добавлять/Удалять к столбцу **значение по умолчанию**

Изменение таблицы

Упрощенный синтаксис оператора **ALTER TABLE (Oracle)**:

```
ALTER TABLE [схема.] имя_таблицы
   {ADD определение_столбца}
 | {MODIFY [COLUMN] определение_столбца}
 | {DROP [COLUMN] имя_столбца)
 | {ADD [CONSTRAINT] ограничение_на_таблицу)
 | {DROP CONSTRAINT имя_ограничения);
```

Изменение таблицы

Упрощенный синтаксис оператора **ALTER TABLE**

```
ALTER TABLE имя таблицы
   {ADD определение столбца}
 | {ALTER COLUMN имя столбца  тип данных
   [NULL | NOT NULL]}
 | {DROP COLUMN имя столбца)
 | {ADD определение ограничения на
таблицу)
 | {DROP [CONSTRAINT] имя ограничения);
```

Переименование таблицы

- Простейший синтаксис переименования таблицы:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

- Пример переименования таблицы:

```
ALTER TABLE suppliers
RENAME TO vendors;
```

Добавление столбца в таблицу

Добавление столбца в таблицу:

```
ALTER TABLE Salespeople
ADD fname CHAR(10);
```

- Синтаксис определения столбца такой же, как в операторе **CREATE TABLE**
- Если в таблице есть строки (т. е. она непустая), столбец добавляется к ним со значением **NULL**
- Новый столбец будет последним столбцом таблицы

Добавление нескольких столбцов

- Добавление сразу нескольких столбцов в таблицу:

```
ALTER TABLE Salespeople
ADD (fname CHAR(10),
    email VARCHAR2(15) );
```

Изменение столбцов

- Изменение типа данных одного столбца:

```
ALTER TABLE Salespeople
MODIFY fname VARCHAR2(50);
```

- Нескольких столбцов:

```
ALTER TABLE Salespeople
MODIFY (fname VARCHAR2(50),
        email VARCHAR2(50));
```

Удаление столбцов

- Удаление столбца из таблицы

```
ALTER TABLE Salespeople
DROP COLUMN fname;
```

Переименование столбцов

- С версии **Oracle 9i** стало доступно переименование столбцов:

```
ALTER TABLE Salespeople
RENAME COLUMN fname TO surname;
```

Удаление таблицы

Удаление таблицы выполняется в два этапа:

- Сначала необходимо удалить из таблицы все данные, используя оператор **DELETE** *(необязательный этап)*
- Затем можно аннулировать определение таблицы с помощью оператора **DROP TABLE**

После удаления таблицы ее имя перестает распознаваться и любые операции с ней становятся невозможны.

Удаление таблицы

Синтаксис оператора **DROP TABLE**:

```
DROP TABLE
имя_таблицы;
```

Ограничение значений данных

Ограничения (constraints) — это элементы определения таблицы, ограничивающие значения, которые можно вводить в ее столбцы (поддержка целостности)

Объявление ограничений

При создании (а также при изменении) таблицы можно наложить ограничения на значения, которые разрешается вводить в ее столбцы:

- **Ограничения на столбец** (column constraints) действуют только на отдельные столбцы
- **Ограничения на таблицу** (table constraints) применяются к группам из одного и более столбцов

Типы ограничений

- **NOT NULL** — исключение NULL-значений (обязательность значений) (только для отдельного столбца!)
- **PRIMARY KEY** — указание первичного ключа
- **UNIQUE** — обеспечение уникальности значений
- **CHECK** — проверка значений столбцов (условие на значение)
- **FOREIGN KEY** и **REFERENCES** — обеспечение ссылочной целостности для группы столбцов таблицы или отдельного столбца

Объявление ограничений

Синтаксис оператора **CREATE TABLE** с указанием ограничений:

```
CREATE TABLE  имя таблицы
({имя столбца   тип данных
  [ограничение на столбец]}...,
 [ограничение на таблицу
  (имя столбца .,..).,..]
);
```

Исключение NULL-значений

NULL — это неопределенное значение, которым отмечаются ячейки таблицы, **не имеющие значения**.

Использование ограничения **NOT NULL**:

```
CREATE TABLE  Salespeople
(snum  NUMBER(10)  NOT NULL,
 sname CHAR(10) NOT NULL,
 city  CHAR(10),
 comm  NUMBER(18,2));
```

Указание первичного ключа

Использование ограничения **PRIMARY KEY** для определения одного столбца в качестве первичного ключа таблицы:

```
CREATE TABLE  Salespeople
(snum  NUMBER(10) PRIMARY KEY,
 sname CHAR(10) NOT NULL,
 city  CHAR(10),
 comm  NUMBER(18,2));
```

Указание первичного ключа

Использование ограничения **PRIMARY KEY** для определения группы столбцов в качестве составного первичного ключа

```
CREATE TABLE Namefield
(firstname CHAR(10),
 lastname  CHAR(10),
 city      CHAR(10),
 PRIMARY KEY (firstname, lastname));
```

Обеспечение уникальности значений

Отличия между ограничениями **UNIQUE** и **PRIMARY KEY**:

- Таблица может содержать ограничение **PRIMARY KEY** только для одного столбца или одной группы столбцов в отличие от **UNIQUE**
- Столбцы с **PRIMARY KEY** не могут содержать **NULL**, а для **UNIQUE** это допустимо
- По-разному взаимодействуют с ограничением **FOREIGN KEY**

Обеспечение уникальности значений

Обеспечение уникальности значений для отдельного столбца с помощью **UNIQUE**:

```
CREATE TABLE  Salespeople
(snum  NUMBER(10)  PRIMARY KEY,
 sname CHAR(10) NOT NULL UNIQUE,
 city  CHAR(10),
 comm  NUMBER(18,2));
```

Обеспечение уникальности значений

Обеспечение уникальности значений для группы столбцов с помощью **UNIQUE**:

```
CREATE TABLE  Salestotal
(snum    NUMBER(10) NOT NULL,
 odate   DATE       NOT NULL,
 totamt  NUMBER(18,2),
 UNIQUE (snum, odate));
```

Проверка значений столбцов

Использование ограничения **CHECK** для проверки значений отдельного

```
CREATE TABLE  Salespeople
(snum  NUMBER(10)  PRIMARY KEY,
 sname CHAR(10) NOT NULL  UNIQUE,
 city  CHAR(10),
 comm  NUMBER(18,2)  CHECK (comm < 1));
```

Любая попытка занести в этот столбец значение, которое делает предикат

Проверка значений столбцов

Использование **CHECK** для задания набора допустимых для столбца значений:

```
CREATE TABLE  Salespeople
(snum  NUMBER(10) PRIMARY KEY,
 sname CHAR(10) NOT NULL  UNIQUE,
 city  CHAR(10) CHECK (city IN
  ('London', 'New York', 'Moscow')),
 comm  NUMBER(18,2)  CHECK (comm < 1));
```

Проверка значений столбцов

Использование ограничения **CHECK** для проверки значений нескольких столбцов:

```
CREATE TABLE  Salespeople
(snum  NUMBER(10) PRIMARY KEY,
 sname CHAR(10) NOT NULL  UNIQUE,
 city  CHAR(10),
 comm  NUMBER(18,2),
 CHECK (comm < .15 OR city = 'Moscow'));
```

Просмотр ограничений таблицы

- Используйте системное представление *user_constraints*

```
SELECT * FROM user_constraints
WHERE table_name =
'ИМЯ_ТАБЛИЦЫ';
```

Именование ограничений

Пример именованного ограничения:

```
CREATE TABLE  Salespeople
(snum NUMBER(10) PRIMARY KEY,
 sname CHAR(10) NOT NULL  UNIQUE,
 city  CHAR(10),
 comm  DECIMAL(18,2),
 CONSTRAINT LuckyMoscow CHECK
  (comm < .15 OR city = 'Moscow'));
```

Правила именований ограничений

- Используйте ключевое слово **CONSTRAINT** для именования ограничений
- Имя ограничения должно быть уникальным среди всех принадлежащих вам имен ограничений
- **Старайтесь всегда давать имена ограничениям**
- Если вы не специфицируете имя ограничения, ORACLE сам назначает имя

Добавление/удаление ограничений

- Добавление именованного ограничения :

```
ALTER TABLE Salespeople
      ADD CONSTRAINT salespeople_uk
            UNIQUE sname;
```

- Удаление именованного ограничения из

```
ALTER TABLE Salespeople
      DROP CONSTRAINT LuckyMoscow;
```

Поддержание ссылочной целостности

SQL поддерживает ссылочную целостность с помощью ограничения **FOREIGN KEY**:

- сужает диапазон вводимых значений, чтобы внешний ключ и его родительский ключ удовлетворяли принципам ссылочной целостности
- отбрасывает значения, которые отсутствуют в родительском ключе
- влияет на возможность изменения и удаления значений родительского ключа

Объявление внешних ключей

Синтаксис ограничения **FOREIGN KEY**, применяемого к таблице:

```
FOREIGN KEY список столбцов
    REFERENCES таблица [список столбцов]
```

- синтаксис можно использовать как в операторе **CREATE TABLE**, так и в **ALTER TABLE**
- Два списка столбцов — для внешнего и родительского ключей — должны быть **совместимы**

Объявление внешних ключей

Использование ограничения **FOREIGN KEY**, применяемого к столбцу:

```
CREATE TABLE  Customers
(cnum    NUMBER(10) PRIMARY KEY,
 cname   CHAR(10),
 city    CHAR(10),
 rating  NUMBER(10),
 snum    NUMBER(10) REFERENCES
           Salespeople(snum)
);
```

Объявление внешних ключей

Использование ограничения **FOREIGN KEY**, применяемого к таблице:

```
CREATE TABLE  Customers
(cnum    NUMBER(10) PRIMARY KEY,
 cname   CHAR(10),
 city    CHAR(10),
 rating  NUMBER(10),
 snum    NUMBER(10),
 FOREIGN KEY (snum)
          REFERENCES Salespeople(snum));
```

Объявление внешних ключей

В ограничении **FOREIGN KEY** можно опустить список столбцов родительского ключа, если этот ключ определен с помощью ограничения **PRIMARY KEY**:

```
CREATE TABLE  Customers
(cnum    NUMBER(10)  PRIMARY KEY,
 cname   CHAR(10),
 city    CHAR(10),
 rating  NUMBER(10),
 snum NUMBER(10) REFERENCES Salespeople);
```

Условия создания REFERENCES

При задании **REFERENCES** должны выполняться два условия:

1. Родительская таблица должна быть создана первой
2. Колонка родительской таблицы, на которую ссылается внешний ключ должна быть **UNIQUE** или **PRIMARY KEY**.
3. Таблица может ссылаться на саму себя (выступать родительской для себя самой)

Использование внешних ключей

Внешний ключ может ссылаться на свою собственную таблицу:

```
CREATE TABLE Employees
(empno    NUMBER(10)  PRIMARY KEY,
 name     CHAR(10) NOT NULL,
 manager NUMBER(10)  REFERENCES Employees);
```

Пример таблицы Employee

Empno	Name	Manager
1003	Terrence	2007
2007	Atali	NULL
1688	McKenna	1003
2002	Collier	2007

Действия, выполняемые по ссылке

Поддержка ссылочной целостности с помощью **действий, выполняемых по ссылке** (referential triggered actions)

Влияния изменений в родительских ключах на внешние ключи:

- Оператор **UPDATE** — обновление значений родительского ключа
- Оператор **DELETE** — удаление значений родительского ключа

Действия, выполняемые по ссылке

По стандарту SQL разрешается независимо изменять поведение операторов **UPDATE** и **DELETE**

Режимы обновления и удаления:

- CASCADE — каскадное обновление или удаление
- SET NULL — установка NULL-значений
- SET DEFAULT — установка значений по умолчанию
- NO ACTION — ограничение обновления или удаления

Действия, выполняемые по ссылке

Синтаксис по стандарту SQL для указания действий, выполняемых по ссылке:

```
[ ON UPDATE { CASCADE
  | SET NULL
  | SET DEFAULT
  | NO ACTION } ]
[ ON DELETE { CASCADE
  | SET NULL
  | SET DEFAULT
  | NO ACTION } ]
```

Ссылочные действия в Oracle

В **Oracle** допустимо только три варианта ссылочных действий для **DELETE**:

- ON DELETE NO ACTION (не явно, по умолчанию)
- ON DELETE CASCADE
- ON DELETE SET NULL

И только одно для UPDATE:

- ON UPDATE NO ACTION (не явно, по умолчанию)

Действия, выполняемые по ссылке

Пример установки режима каскадного удаления:

```
CREATE TABLE  Customers
(cnum    NUMBER(10) PRIMARY KEY,
 cname   CHAR(10),
 city    CHAR(10),
 rating  NUMBER(10),
 snum NUMBER(10) REFERENCES Salespeople
                 ON DELETE CASCADE);
```