



Тема 10. Язык запросов SQL. Введение

Лекция 10

* План лекции

1. Общая характеристика SQL.
2. Стандарты SQL.
3. Реализации SQL в современных СУБД.
SQL-серверы.



Structured Query Language

SQL - Structure Query Language

Предшественником SQL был язык **SEQUEL**
(Structure English Query Language) .

Близок к реляционному исчислению
кортежей

Был разработан в середине 1970 году
(IBM) - System R. Первой коммерческой
системой, в которой был реализован этот
язык, была система Oracle (1979 г.) .

SQL – Structured Query Language

- * **SQL - это структурированный язык запросов** к реляционным базам данных (БД).
- * **SQL - декларативный язык**, основанный на операциях реляционной алгебры.
- * Стандарты SQL, определённые Американским национальным институтом стандартов (ANSI):
 - ✓ SQL-1 (SQL/89) - первый вариант стандарта.
 - ✓ SQL-2 (SQL/92) - основной расширенный стандарт.
 - ✓ SQL-3 (SQL/1999, SQL/2003) - относится к объектно-реляционной модели данных.
- * Подмножества языка SQL:
 - ✓ **DDL** (Data Definition Language) - команды создания/изменения/удаления объектов базы данных (*create/alter/drop*);
 - ✓ **DML** (Data Manipulation Language) - команды добавления/модификации/удаления данных (*insert/update/delete*), а также команда извлечения данных *select*;
 - ✓ **DCL** (Data Control Language) - команды управления данными

*Язык SQL

Structured Query Language – стандартный язык управления реляционными базами данных с архитектурой клиент-сервер.

Целью разработки было создание простого непроцедурного языка, которым мог бы воспользоваться любой пользователь, даже не имеющий навыков программирования.

Правило №5 д-ра Кодда (из правил-требований к реляционной модели) гласит: язык доступа к данным должен быть единственным способом доступа к данным в реляционной СУБД.

* Язык SQL- стандарты

Поскольку к началу 1980-х годов существовало несколько вариантов СУБД от разных производителей, причём каждый из них обладал собственной реализацией языка запросов, было принято решение разработать стандарт языка, который будет гарантировать переносимость ПО с одной СУБД на другую (при условии, что они будут поддерживать этот стандарт).

Основные стандарты:

1986 г. - SQL-86 (SQL1)

1992 г. - SQL2

1999 г. – SQL3

2003 г. – SQL2003



Язык SQL – Преимущества

Независимость от конкретной СУБД

диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД.

Наличие стандартов

Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка..

Декларативность

С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его

* Язык SQL - Недостатки

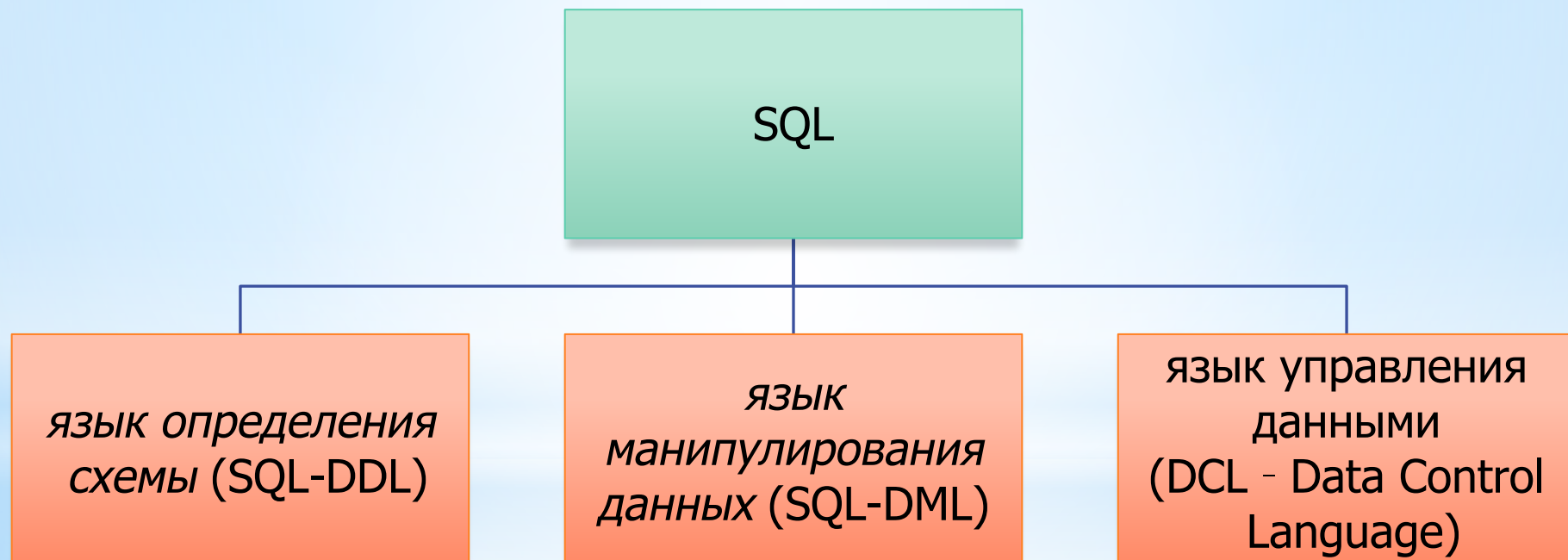
Несоответствие реляционной модели данных Создатели реляционной модели данных [Эдгар Кодд](#), [Кристофер Дейт](#) и их сторонники указывают на то, что SQL не является истинно реляционным языком.

Сложность Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста.

Отступления от стандартов Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, [Oracle](#), [Sybase](#), [Microsoft](#), [MySQL AB](#)), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД. Таким образом, появляются специфичные для каждой конкретной СУБД диалекты языка SQL.

Сложность работы с иерархическими структурами Ранее диалекты SQL большинства СУБД не предлагали способа манипуляции древовидными структурами. В настоящее время в ANSI стандартизована рекурсивная конструкция WITH из диалекта SQL [DB2](#).

* Подмножества языка SQL



SQL

```
graph TD; SQL[SQL] --- I[Интерактивный]; SQL --- S[Статический]; SQL --- D[Динамический]; S --- V[встроенный]; S --- M[модульный];
```

Интерактивный

Статический

Динамический

встроенный

модульный

* DDL - data definition language (язык определения данных)

включает всевозможные команды создания (**CREATE**), удаления (**DROP**) и изменения структуры (**ALTER**) объектов, таких, как таблицы (**TABLE**), представления (**VIEW**), триггеры (**TRIGGER**), пользователи (**USER**) и т.п.

Пример создания таблицы “Организации”:

```
CREATE TABLE k_firm
(firm_num NUMERIC(6) PRIMARY KEY,
firm_name VARCHAR(100) NOT NULL,
firm_addr VARCHAR(100)
);
```

**Политики ссылочной целостности*

- Политика **IGNORE** означает, что мы не предусматриваем никаких проверок и ограничений.
- Политика **RESTRICT** действует, когда мы применяем ограничения внешних ключей.
- При использовании политики **CASCADE** мы должны предусмотреть собственную программную обработку, т.е. при изменении родительских таблиц вносить изменения в дочерние таблицы программным образом.
- Политика **SET DEFAULT** состоит в том, что при изменении данных в родительских таблицах дочерним таблицам назначаются значения по умолчанию. Например, при удалении отдела мы можем записать его сотрудников в некоторый другой отдел, который мы считаем отделом по умолчанию.
- Политика **SET NULL** похожа на предыдущую политику, только вместо значений по умолчанию мы назначаем NULL-значения.

* Declarative Referential Integrity (декларативная ссылочная целостность)

На уровне определения таблиц осуществляется декларативная политика ссылочной целостности с помощью внешних ключей. Она требует, чтобы в поле *внешнего ключа* можно было вводить только такие значения *первичного ключа*, которые присутствуют в родительской таблице.

Пример создания таблицы “Договоры”:

```
CREATE TABLE k_contract
( contract_num NUMERIC(6) PRIMARY KEY,
  contract_date DATETIME,
  contract_type CHAR(1) CHECK (contract_type IN
('A', 'B', 'C')),
  firm_num NUMERIC(6) NOT NULL,
CONSTRAINT fk_contract_firm_num FOREIGN KEY
(firm_num) REFERENCES k_firm (firm_num) );
```


*** DML - data manipulation language (язык манипулирования данными)**

включает команды **INSERT, DELETE, UPDATE.**

Команда добавления строк в таблицу:

```
INSERT [INTO] имя_таблицы [ (список_полей) ]  
VALUES (список_значений)
```

Команда обновления строк таблицы:

```
UPDATE имя_таблицы SET поле1=выражение1  
[, ... , полеN=ВыражениеN] [WHERE условие]
```

Команда удаления строк таблицы:

```
DELETE [FROM] имя_таблицы [WHERE условие]
```

*DQL - data query language (язык запросов к данным)

содержит огромную команду **SELECT**, имеющую возможности:

- выборки из одной или из нескольких таблиц,
- использования условий отбора,
- сортировки,
- использования подзапросов,
- использования агрегатных функций,
- группировки,
- объединения запросов.

*CCL - cursor control language (язык управления курсорами)

Cursor – **current set of record** - временный набор записей, позволяющий обрабатывать каждую запись по отдельности. Необходимость использования курсоров возникла потому, что команды изменения данных (UPDATE, DELETE) применяются к таблице целиком и поэтому являются достаточно “грубыми” для разнообразной “тонкой” работы.

Стандартные операции по работе с курсором:

- объявление курсора
`DECLARE имя_курсора CURSOR FOR SELECT команда`
- открытие курсора: `OPEN имя_курсора`
- получение значений из текущей строки и передвижение указателя на следующую строку:
`FETCH имя_курсора INTO переменные`
- закрытие курсора: `CLOSE имя_курсора`

*TPL - transaction processing language (язык проведения транзакций)

Транзакция – группа команд языка SQL, которая либо выполняется полностью, либо не выполняется вообще.

Стандартные команды для работы с транзакциями:

BEGIN TRAN – начало транзакции,

ROLLBACK TRAN – откат, отмена транзакции; все изменения, сделанные с начала транзакции, будут отменены,

COMMIT TRAN – завершение, подтверждение транзакции; все изменения, сделанные с начала транзакции, будут зафиксированы.

До момента подтверждения транзакции все измененные данные записываются в журнал транзакций, и только после фиксации транзакции данные переносятся собственно в таблицы.

* Уровни изолированности транзакций:

Serializable – самый надежный, но и самый медленный уровень изолированности, транзакции выполняются последовательно, друг за другом.

Repeatable read – при повторном чтении данных результат будет точно таким же, как и при первом чтении, даже если данные были изменены.

Read committed – допускается читать только данные завершенных транзакций.

Read uncommitted - допускается читать “грязные данные”, т. е., данные незавершенных транзакций.

*DCL - data control language (язык управления данными)

содержит команды предоставления (**GRANT**) и отнимания (**REVOKE**) прав доступа, а также запрета доступа (**DENY**).

Примеры:

предоставление прав на выборку и изменение данных в таблице k_contract пользователю public:

```
GRANT SELECT, UPDATE ON k_contract TO public
```

запрет удаления данных из таблицы k_contract пользователю public:

```
DENY DELETE ON k_contract FROM public
```

* Виды запросов

- * Поисковые

- * Корректирующие

- * включение новой записи (INSERT),

- * обновление отдельных полей (UPDATE),

- * удаление записи или группы записей (DELETE).

*Поисковый запрос

```
SELECT <список колонок, включаемых в ответ>  
FROM <список таблиц>  
WHERE <условие>
```

пример:

```
SELECT * FROM kadr WHERE vozr = 40 AND pol =  
    «М»;
```

Оператор SELECT оперирует над множествами и результатом обработки в общем случае является множество строк. К этим множествам могут быть применены теоретико-множественные операции объединение (UNION), пересечение (INTERSECTION), разность (DIFFERENCE, MINUS, EXCEPT) и другие. В разных реализациях языка SQL наборы теоретико-множественных операций различаются

*Язык SQL позволяет
запрашивать вычисляемые
значения

пример:

```
SELECT naimprod, datapost,  
       kolv*cena
```

```
FROM postypl
```


Возможна подгруппировка данных с целью получения подитогов или других обобщающих величин.

В стандарт SQL-92 включены следующие агрегатные функции:

✓ *count* - подсчет,

✓ *avg* - среднее,

✓ *sum* - сумма,

✓ *max* - максимум,

✓ *min* - минимум.

* Использование трехзначной логики

* Истина (True)

* Ложь (False)

* Неопределенное значение (NULL)

*Стандарты SQL

В 1983 году Международная организация по стандартизации (ISO) и Американский национальный институт стандартов (ANSI) приступили к разработке стандарта языка SQL. Стандарт SQL был впервые опубликован в 1986 г. (SQL-86)- обеспечивал минимальную функциональность.

Обновлялся:

1989 - (SQL-89) механизм поддержания ссылочной целостности,

1992 - (SQL-2) расширенная функциональность,

1999 - (SQL-3) добавлена поддержка регулярных выражений, рекурсивных запросов, поддержка триггеров, и некоторые объектно-ориентированные возможности,

*Стандарты SQL

2003 - (SQL:2003) введены расширения для работы с XML-данными, оконные функции (применяемые для работы с OLAP-базами данных), генераторы последовательностей и основанные на них типы данных,

2006 - (SQL:2006) функциональность работы с XML-данными значительно расширена. Появилась возможность совместно использовать в запросах SQL и Xquery,

2008 - (SQL:2008) улучшены возможности оконных функций, устранены некоторые неоднозначности стандарта SQL:2003.

*SQL-серверы

Oracle	Oracle Corp.	www.oracle.com
MySQL	Oracle Corp.	www.mysql.com
MS SQL Server	Microsoft	www.microsoft.com
Informix	Informix	www.informix.com
Sybase	Sybase	www.sybase.com
DB2	IBM	www.4.ibm.com

*Диалекты SQL

Каждая СУБД имеет свой собственный “диалект” SQL, включающий, кроме основ SQL, команды управления (циклы, условия), функции и прочие средства:

- ORACLE – PL/SQL (Procedural Language/SQL),
 - MS SQL Server – Transact SQL,
 - MySQL – SQL/PSM (SQL/Persistent Stored Module)
- и т.п.

Работа с SQL в СУБД Oracle, MySQL

* Особенности синтаксиса:

- ✓ В командах SQL не различаются прописные и строчные буквы (кроме содержимого символьных строк).
- ✓ Каждая команда может занимать несколько строк и заканчивается символом ';'.
;.
- ✓ Символ и символьная строка заключается в одинарные кавычки:
'А', '2' , 'строка', 'другая строка'
- ✓ Однострочный комментарий начинается с символов '--'.
- ✓ Многострочный комментарий заключается в символы
/* ... */.

* SQL-приложения СУБД Oracle:

- ✓ SQL Work Sheet;

* SQL -приложения СУБД MySQL .

Команды DDL

CREATE - создание объекта.

ALTER - изменения структуры объекта.

DROP - удаление объекта.

Общий вид синтаксиса команд DDL:

create	тип_объекта имя_объекта [параметры];
alter	
drop	

Создание таблиц

```
CREATE TABLE [имя_схемы.]имя_таблицы  
  ( имя_поля  тип_данных [(размер)] [NOT NULL]  
    [DEFAULT выражение]  
    [ограничения_целостности_поля...]  
    ...,  
    [, ограничения_целостности_таблицы .,...]  
  )  
[ параметры ];
```

ограничения_целостности (ОЦ):

```
[CONSTRAINT имя_ОЦ ] название_ОЦ [параметры]
```

* Типы данных

* Символьные типы:

✓ **CHAR** [(длина)] - строка фиксированной длины.

Длина по умолчанию - 1, максимальная длина 2000 байт.

Строка дописывается до указанной длины пробелами.

✓ **VARCHAR2** (длина) - строка переменной длины.

Максимальная длина 4000 байт. Хранятся только значащие символы.

* Числовой тип:

✓ **NUMBER** [(точность[, масштаб])] - используется для представления чисел с заданной точностью.

Точность (максимально допустимое число значащих десятичных цифр) по умолчанию 38, масштаб (количество цифр после десятичной точки) по умолчанию - 0.

number(4) - числа от -999 до 9999

number(8,2) - числа от -99999.99 до 999999.99

* **DATE** - дата и время с точностью до секунды. Занимает 7 байт.

✓ sysdate - функция получения текущих даты и времени.

✓ Тип date поддерживает арифметику дат:

sysdate+1 - завтра

(дата1 - дата2) - количество дней, прошедших между двумя датами

Ограничения целостности

В СУБД MySQL поддерживаются следующие ограничения целостности:

- ✓ уникальность (значений атрибута или комбинации значений атрибутов):

UNIQUE (*имя_атрибута1* [, *имя_атрибута2*,...])

- ✓ обязательность / необязательность:

NOT NULL / **NULL**

- ✓ первичный ключ:

PRIMARY KEY(*имя_атрибута1* [, *имя_атрибута2*,...])

- ✓ внешний ключ:

FOREIGN KEY(*имя_атрибута1* [, *имя_атрибута2*,...])

REFERENCES *имя_таблицы* [(*имя_атрибута1* [, *имя_атрибута2*,...])]

- ✓ условие на значение поля:

CHECK (*условие*)

Например: `check (salary >= 4500)` `check (date2 > date1)`

* Типы данных MySQL

* Символьные типы:

✓ **CHAR** [(длина)] - строка фиксированной длины.

Длина по умолчанию - 1, максимальная длина 255 байт.

Строка дописывается до указанной длины пробелами.

✓ **VARCHAR** (длина) - строка переменной длины.

Максимальная длина 255 байт. Хранятся только значащие символы.

* Числовые типы:

NUMERIC [(точность[, масштаб])] - используется для представления чисел с заданной точностью.

Масштаб по умолчанию - 0.

numeric(4) - числа от -999 до 9999

numeric(8,2) - числа от -99999.99 до 999999.99

MySQL поддерживает все числовые типы данных языка SQL92 по стандартам ANSI/ISO. Они включают в себя типы точных числовых данных (NUMERIC, DECIMAL, INTEGER и SMALLINT) и типы приближенных числовых данных (FLOAT, REAL и DOUBLE PRECISION). Ключевое слово INT является синонимом для INTEGER, а ключевое слово DEC - синонимом для DECIMAL.

* Числовые типы данных MySQL

Таблица 1 – Целочисленные типы данных

Тип	Диапазон	Память (байт)
TINYINT[(M)]	-127..128 или 0..255	1
BIT		1
BOOL		1
SMALLINT[(M)]	-32768..32767 или 0..65535	2
MEDIUMINT[(M)]	-8388608..8388607 или 0..16777215	3
INT[(M)]	$-2^{31}..2^{31} - 1$ или $0..2^{32} - 1$	4
INTEGER[(M)]		4
BIGINT[(M)]	$-2^{63}..2^{63} - 1$ или $0..2^{64} - 1$	8

Таблица 2 – Типы данных с плавающей запятой

Тип	Диапазон	Память (байт)
FLOAT(<i>точность</i>)	зависит от точности	различна
FLOAT[(M, D)]	$\pm 1.175494351E-38$ $\pm 3.402823466E+38$	4
DOUBLE[(M, D)]	$\pm 1.7976931348623157E+308$ $\pm 2.2250738585072014E-308$	8
DOUBLE PRECISION[(M, D)]	$\pm 1.7976931348623157E+308$ $\pm 2.2250738585072014E-308$	8
REAL[(M, D)]	$\pm 1.7976931348623157E+308$ $\pm 2.2250738585072014E-308$	8
DECIMAL[(M, D)]	различный	M + 2
NUMERIC[(M, D)]	различный	M + 2
DEC[(M, D)]	различный	M + 2
FIXED[(M, D)]	различный	M + 2

* Типы данных MySQL: дата и время

Тип	Описание
DATE	Дата в формате ГГГГ-ММ-ДД
TIME	Время в формате ЧЧ:ММ:СС
TIMESTAMP	Дата и время в формате timestamp, выводится в виде ГГГГММДДЧЧММСС
DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС

Величины DATETIME, DATE и TIMESTAMP задаются:

- Как строка в формате 'YYYY-MM-DD HH:MM:SS' ('YYYY-MM-DD') или в формате 'YY-MM-DD HH:MM:SS' ('YY-MM-DD'). Допускается ``облегченный" синтаксис - можно использовать любой знак пунктуации в качестве разделительного между частями разделов даты или времени.

Например, величины '98-12-31 11:30:45', '98.12.31 11+30+45', '98/12/31 11*30*45' и '98@12@31 11^30^45' являются эквивалентными.

- Как строка без разделительных знаков в формате 'YYYYMMDDHHMMSS' ('YYYYMMDD') или в формате 'YYMMDDHHMMSS' ('YYMMDD').
- Как число в формате YYYYMMDDHHMMSS или в формате YYMMDDHHMMSS.
- Как результат выполнения функции, возвращающей дату (например, функции NOW() или CURRENT_DATE).

Недопустимые значения величин DATETIME, DATE или TIMESTAMP преобразуются в значение ``ноль" соответствующего типа величин ('0000-00-00 00:00:00', '0000-00-00', или 0000000000000000).

* Типы данных MySQL: время

MySQL извлекает и выводит величины типа TIME в формате 'HH:MM:SS'.

Величины TIME могут изменяться в пределах от '-838:59:59' до '838:59:59'.

Величины TIME могут быть заданы в различных форматах:

Как строка в формате 'D HH:MM:SS' (следует учитывать, что MySQL пока не обеспечивает хранения дробной части величины в столбце рассматриваемого типа).

Можно также использовать одно из следующих "облегченных" представлений:

HH:MM:SS, HH:MM, D HH:MM, D HH или SS. Здесь D – это дни из интервала значений 0-33.

Как строка без разделителей в формате 'HHMMSS', при условии, что строка интерпретируется как дата.

Как число в формате HHMMSS, при условии, что строка интерпретируется как дата. Например, величина 101112 понимается как '10:11:12'.

Как результат выполнения функции, возвращающей величину, приемлемую в контексте типа данных типа TIME (например, CURRENT_TIME).

Примеры:

'101112' → '10:11:12'

'109712' → '00:00:00'

'8:3:2' → '08:03:02'

'1112' и 1112 → '00:11:12' – крайние правые разряды – секунды!

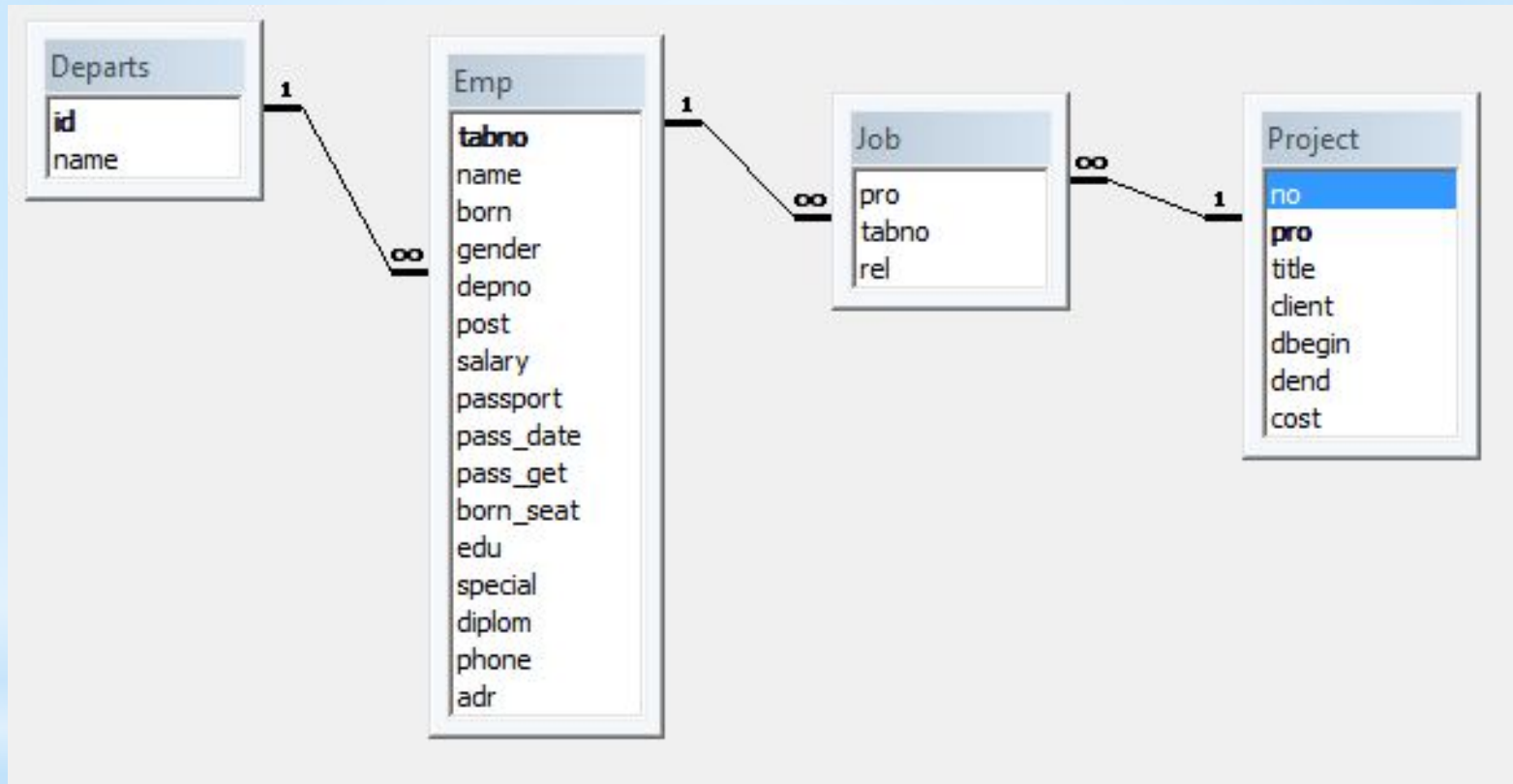
'11:12' → '11:12:00' – двоеточие означает, что крайние левые разряды – часы

* Ограничения целостности

По стандарту ANSI/SQL поддерживаются следующие ограничения целостности:

- ✓ уникальность (значений атрибута или комбинации значений полей):
UNIQUE (*имя_поля1* [, *имя_поля2*,...])
- ✓ обязательность / необязательность:
NOT NULL / **NULL**
- ✓ первичный ключ:
PRIMARY KEY(*имя_поля1* [, *имя_поля2*,...])
- ✓ внешний ключ:
FOREIGN KEY(*имя_поля1* [, *имя_поля2*,...]) **REFERENCES** *имя_таблицы*
[(*имя_поля1* [, *имя_поля2*,...])]
[ON DELETE CASCADE | ON DELETE SET NULL]
- ✓ условие на значение поля:
CHECK (*условие*)
Например: `check (salary >= 4500), check (date2 > date1)`

Пример БД: проектная организация



Departments – отделы,

Emp – сотрудники,

Project – проекты,

Job – участие в проектах.

* Организация связей между таблицами

Связь один-ко-многим: Отделы – Сотрудники

Таблица «Сотрудники» (Emp)

<i>Табельный номер</i>	<i>ФИО сотрудника</i>	<i><u>Отдел</u></i>
023	Волкова Елена Павловна	2
113	Белов Сергей Юрьевич	1
101	Рогов Сергей Михайлович	2
056	Панина Анна Алексеевна	1
...
098	Фролов Юрий Вадимович	9

Таблица «Отделы» (Departs)

<i>Номер отдела</i>	<i>Название отдела</i>
1	Информационный отдел
2	Администрация
3	Отдел кадров
...	...
9	Проектный отдел

«Номер отдела» – первичный ключ в таблице «Отделы»

«Отдел» – внешний ключ в таблице «Сотрудники» к таблице «Отделы»

* Организация связей между таблицами

Связь многие-ко-многим: Проекты – Сотрудники

Таблица «Сотрудники» (Emp)

<i>ФИО</i>	<i>Номер</i>
Волкова Е.П.	023
Белов С.Ю.	113
Рогов С.М.	101
Панина А.А.	056
Фролов Ю.В.	098
...	...

Таблица «Проекты» (Project)

<i>Шифр</i>	<i>Название проекта</i>
23/Н	АИС "Налог"-2
18-К	ИПС "Жители"
09/Р	ГИС "Город"
...	...

Таблица «Участие» (Job)

<i>Участник</i>	<i>Роль</i>	<i>Проект</i>
113	исполнитель	23/Н
101	руководитель	18-К
056	исполнитель	18-К
101	консультант	09/Р
098	руководитель	23/Н
...

В таблице «Участие»:

«Участник» – внешний ключ к таблице «Сотрудники»

«Проект» – внешний ключ к таблице «Проекты»

Пример БД: проектная организация

Emp – сотрудники:

tabno – табельный номер сотрудника, первичный ключ;

name – ФИО сотрудника, обязательное поле;

born – дата рождения сотрудника, обязательное поле;

gender – пол сотрудника, обязательное поле;

depno – номер отдела, обязательное поле, внешний ключ;

post – должность сотрудника;

salary – оклад, больше МРОТ;

passport – серия и номер паспорта, уникальный обязательный атрибут;

pass_date – дата выдачи паспорта, обязательное поле;

pass_get – кем выдан паспорт, обязательное поле;

born_seat – место рождения сотрудника;

edu – образование сотрудника;

special – специальность по образованию;

diplom – номер диплома;

phone – телефоны сотрудника;

adr – адрес сотрудника;

edate – дата вступления в должность, обязательное поле.

Пример БД: проектная организация

Departs – отделы:

did – номер отдела, первичный ключ;

name – название отдела, обязательное поле.

Project – проекты:

No – номер проекта, первичный ключ;

title – название проекта, обязательное поле;

pro – краткое название проекта, обязательное уникальное поле;

client – заказчик, обязательное поле;

dbegin – дата начала выполнения проекта, обязательное поле;

dend – дата завершения проекта, обязательное поле;

cost – стоимость проекта, обязательное поле.

Job – участие в проектах:

pro – краткое название проекта, внешний ключ;

tabNo – номер сотрудника, участвующего в проекте, внешний ключ;

rel – роль сотрудника в проекте; может принимать одно из трех значений: 'исполнитель', 'руководитель', 'консультант'.

Первичный ключ – комбинация полей **pro** и **tabNo**.

Создание таблиц БД проектной организации

Таблица «Отделы» (Depart):

```
create table depart (did number(4) constraint pk_depart PRIMARY KEY,  
    name varchar(100) not null  
);
```

Таблица «Сотрудники» (Emp):

```
create table emp (tabno number(6) constraint pk_emp PRIMARY KEY,  
    name varchar(100) not null,  
    born date not null,  
    gender char not null,  
    depno number(4) not null constraint fk_depart REFERENCES depart,  
    post varchar(50) not null,  
    salary number(8,2) not null constraint check_sal check (salary > 4630),  
    passport char(10) not null constraint passp_uniq UNIQUE,  
    pass_date date not null,    pass_get varchar(100) not null,  
    born_seat varchar(100),    edu varchar(30),  
    special varchar(100),    diplom varchar(40),  
    phone varchar(30),    adr varchar(80),  
    edate date not null default trunc(sysdate),  
    chief number(6) constraint fk_emp REFERENCES emp  
);
```


Создание таблиц БД проектной организации

Таблица «Проекты» (Project):

```
create table project (No number(5) constraint pk_project primary key,  
    title varchar(200) not null,  
    pro varchar(15) not null constraint pro_uniq unique,  
    client varchar(100) not null,  
    dbegin date not null,  
    dend date not null,  
    cost number(9)  
);
```

Таблица «Участие в проектах» (Job):

```
create table job (pro varchar(15) not null references project (abbr),  
    tabNo number(6) not null references emp,  
    rel varchar(20) default 'исполнитель',  
    primary key (tabno, pro),  
    check ( rel IN ('исполнитель', 'руководитель', 'консультант') )  
);
```


Подмножество команд DML

* **INSERT** - добавление строк в таблицу.

✓ Добавляет одну или несколько строк в указанную таблицу.

* **UPDATE** - изменение данных.

✓ Изменяет значения одного или нескольких полей в записях указанной таблицы.

✓ Можно указать условие, по которому выбираются обновляемые строки.

✓ Если условие не указано, обновляются все строки таблицы.

✓ Если ни одна строка не удовлетворяет условию, ни одна строка не будет обновлена.

* **DELETE** - удаление строк из таблицы.

✓ Удаляет одну или несколько строк из таблицы.

✓ Можно указать условие, по которому выбираются удаляемые строки.

✓ Если условие не указано, удаляются все строки таблицы.

Добавление данных

INSERT - добавление строк в таблицу:

```
INSERT INTO имя_таблицы [(список_полей_таблицы)]  
    { VALUES (список_выражений) | запрос };
```

Примеры:

-- Добавить в таблицу "Отделы" новую запись (все поля):

```
insert into depart  
values(7, 'Договорной отдел');
```

-- Добавить в таблицу "Сотрудники" новую запись (не все поля):

```
insert into emp (tabno, name, born, gender, depno, passport,  
pass_date_pass_get,  
post, salary, phone)  
values( 301, 'САВИН АНДРЕЙ ПАВЛОВИЧ', to_date('11.07.1969',  
'dd.mm.yyyy'),  
'М', 5, '4405092876', to_date('15.02.1999', 'dd.mm.yyyy'),  
'ОВД "Митино" г.Москвы', 'программист', 38050, '121-34-11');
```

Замечание: значение по умолчанию используется только тогда, когда значение поля не вводится в явном виде.

*Изменение данных

UPDATE - изменение данных:

UPDATE *имя_таблицы*

SET *имя_поля1* = *выражение1* [, *имя_поля2* = *выражение2*,...]

[**WHERE** *условие*];

Примеры:

-- Изменить статус сотрудника Бобкова Л.П., табельный номер 74, по отношению к проекту 30. "Система автоматизированного управления предприятием":

update job

set rel = 'консультант'

where tabno = 74 and **pro** = 30;

-- Перевести сотрудника Жаринова А.В., табельный номер 68, на должность ведущего программиста и повысить оклад на три тысячи рублей:

update emp

set post = 'ведущий программист', **salary** = **salary**+3000

where tabno = 68;

Удаление данных

DELETE - удаление строк из таблицы:

```
DELETE FROM имя_таблицы  
[ WHERE условие ];
```

Примеры.

-- Удалить сведения о том, что сотрудник Афанасьев В.Н.,
табельный номер 147, участвует в проектах:

```
delete from job  
where tabno=147;
```

-- Удалить сведения о сотруднике Афанасьеве В.Н.,
табельный номер 147:

```
delete from emp  
where tabno = 147;
```

Замечание: отменить удаление данных можно командой
ROLLBACK;

Изменение структуры таблицы

Оператор **ALTER TABLE**

обеспечивает возможность изменять структуру существующей таблицы. Например, можно добавлять или удалять столбцы, создавать или уничтожать индексы или переименовывать столбцы либо саму таблицу.

Пример.

-- В таблице **emp** переименовать столбец **post** в **position**:

```
ALTER TABLE emp CHANGE COLUMN post position  
varchar(50) not null;
```

Спасибо за внимание!