Лекция №7. Выборка данных. Соединения.

Управление данными

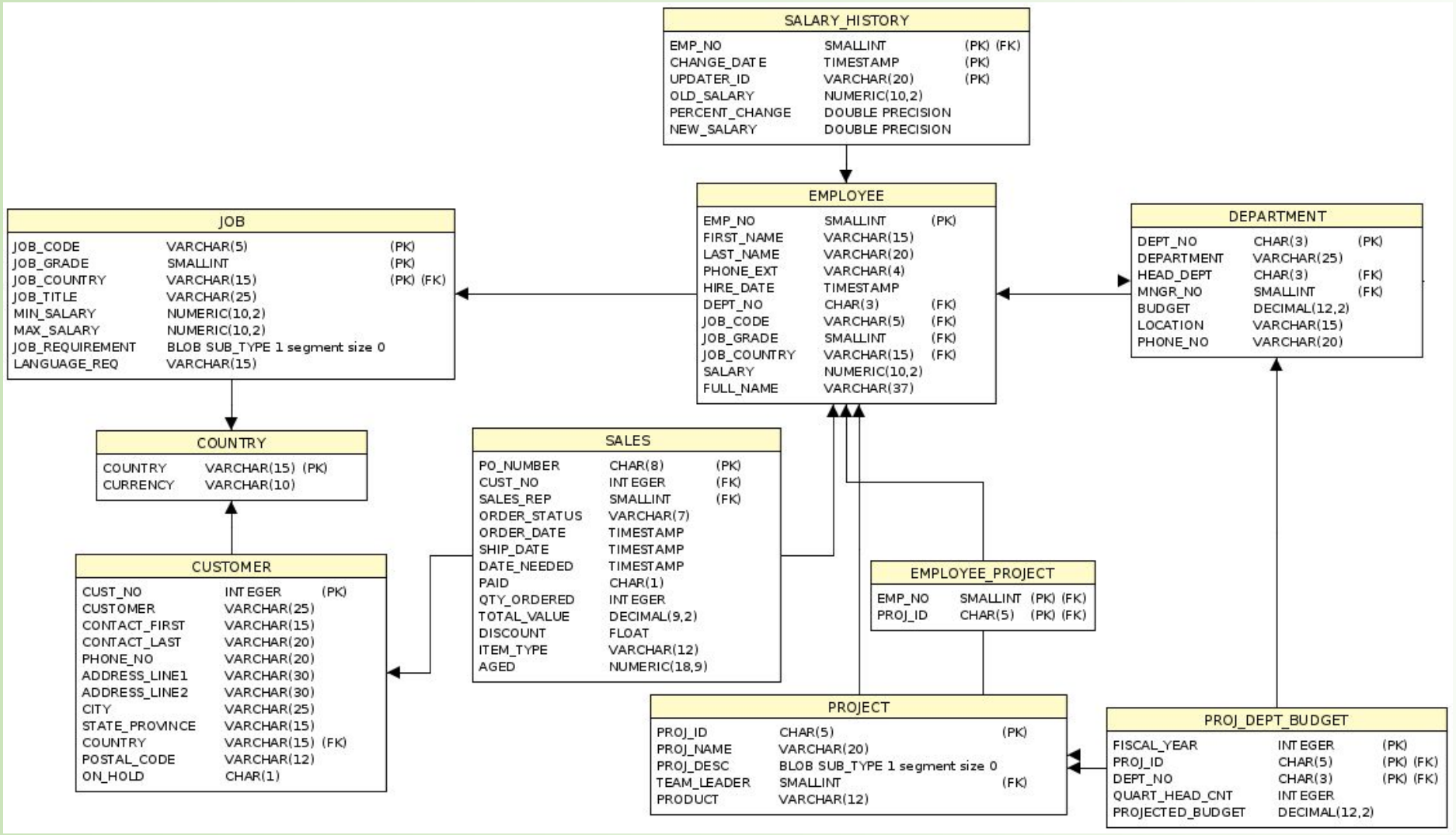
Выборка данных

- Оператор SELECT - оператор DML (Data Manipulation Language)
- Самый сложный и мощный оператор
  - Выбирает данные из нескольких таблиц
  - Объединяет или фильтрует данные
  - Агрегирует
  - Сортирует
  - Возвращает данные в виде множества строк заданной структуры (таблица)

Синтаксис оператора SELECT

```
[WITH [RECURSIVE] <CTE> [, <CTE> ...]]
  SELECT [FIRST <значение>] [SKIP <значение>] [DISTINCT | ALL]
    <выходное поле> [, <выходное поле>]
  FROM <источники> [<соединяемые источники>]
  [WHERE <условие выборки>]
  [GROUP BY <условие группирование выбранных данных>
[HAVING <условие выборки>]]
[UNION [DISTINCT | ALL] <другой набор данных>] [PLAN <выражение для плана
поиска>]
[ORDER BY <выражение для порядка выборки>]
[ ROWS <m> [TO <n>]
    | [OFFSET <n> {ROW | ROWS}] [FETCH {FIRST | NEXT} [<m>] {ROW | ROWS} ONLY]
]
  [FOR UPDATE [OF <имя столбца> [, <имя столбца>]...] [WITH LOCK]
  [INTO [:]<переменная> [,[:]<переменная> ... ]]
```

Структура БД примера (employee.fdb)



Простые примеры запроса SELECT

- `SELECT * FROM COUNTRY;`

- `SELECT CURRENCY FROM COUNTRY WHERE COUNTRY='Russia';`

- `SELECT COUNT(*) FROM CUSTOMER;`

- `SELECT E.FULL_NAME FROM EMPLOYEE E ORDER BY E.FULL_NAME;`

Список выбора

- Список полей или выражений, после слова SELECT
- Определяет состав и тип полей результата
- SELECT <поле>, <поле>, ... FROM T;
- <поле> может быть:
  - *
  - ТАБЛИЦА.ПОЛЕ
  - ПРОЦЕДУРА.ВЫХОДНОЕ_ПОЛЕ
  - Константа
  - NULL
  - Выражение
  - Конструкция CASE
  - NEXT VALUE FOR
  - Любое выражение, возвращающее единственное значение
- Вместо ТАБЛИЦА можно указать псевдоним, представление

Примеры списков выбора

- `SELECT * FROM RDB$DATABASE;`
- `SELECT CUSTOMER.CUSTOMER, CUSTOMER.PHONE_NO, CITY FROM CUSTOMER;`
- `SELECT LAST_NAME, SALARY * 12 AS ANNUAL_SALARY FROM EMPLOYEE;`
- `SELECT iif(CITY IS NULL, 'Murom', CITY) FROM CUSTOMER;`
- `SELECT UPPER(COUNTRY) FROM COUNTRY;`

Ограничения выборки

- После SELECT могут быть указаны ключевые слова:
  - FIRST - означает выбрать указанное количество первых записей
  - SKIP - означает пропустить указанное количество записей
  - DISTINCT - означает, что выбираемые строки должны отличатся друг от друга. Дубликаты будут исключены. Может требовать сортировку.
  - ALL - означает, что надо выбрать все строки. По умолчанию.
- Кроме FIRST и SKIP может использоваться ORDER BY вместе с предложениями:
  - OFFSET - аналогично SKIP, но входит в стандарт
  - FETCH - аналогично FIRST, но входит в стандарт
- Рекомендуется использовать именно OFFSET и FETCH

Примеры

- `SELECT FIRST 10 CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC`
- `SELECT CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC FETCH FIRST 10 ROWS ONLY`

- `SELECT SKIP 10 CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC`
- `SELECT CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC OFFSET 10 ROWS`

- `SELECT SKIP ((SELECT COUNT(*) - 10 FROM CUSTOMER)) CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC`
- `SELECT CUST_NO FROM CUSTOMER ORDER BY FIRST_NAME ASC OFFSET ((SELECT COUNT(*) - 10 FROM CUSTOMER)) ROWS`

Выражение FROM

- Определяет источники, из которых будут отобраны данные:
  - Таблицы - простейший случай
  - Представление
  - Хранимая процедура
  - Производная таблица
  - Общее табличное выражение
- Источники можно комбинировать используя:
  - Декартово произведение - через запятую
  - Операторы соединения (JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN)
- Простая выборка:
  - `SELECT * FROM RDB$DATABASE;`

Алиасы источников

- Источникам можно давать алиасы (псевдонимы)
- Если источнику задан алиас, то надо использовать везде его, а не имя таблицы

- Корректно:
  - `SELECT LAST_NAME FROM CUSTOMER;`
  - `SELECT CUSTOMER.LAST_NAME FROM CUSTOMER;`
  - `SELECT LAST_NAME FROM CUSTOMER C;`
  - `SELECT C.LAST_NAME FROM CUSTOMER C;`
- Не корректно:
  - `SELECT ~~CUSTOMER~~.LAST_NAME FROM CUSTOMER C;`

Выборка из производной таблицы

- Производная таблицы - это команда SELECT заключенная в круглые скобки
- Может сопровождаться псевдонимами таблицы и полей
- Тривиальный пример:
  - `SELECT **DBINFO**.DESCR, **DBINFO**.DEF_CHARSET FROM *(SELECT * FROM RDB$DATABASE)* ***DBINFO*** *(DESCR, REL_ID, SEC_CLASS, DEF_CHARSET)*`
  - Алиас производной таблицы выделен жирным.
  - После него в скобках идут алиасы выбираемых полей, которые используются в основном запросе.
- Производные таблицы могут быть вложенными
- Каждый столбец должен иметь имя или присвоенный псевдоним
- Список псевдонимов столбцов опциональный, но если есть то полный

Пример выборки из производной таблицы

- Допустим есть таблица с коэффициентами квадратных уравнений:
  - ```
    CREATE TABLE coeffs (
        a DOUBLE PRECISION NOT NULL,
        b DOUBLE PRECISION NOT NULL,
        c DOUBLE PRECISION NOT NULL,
        CONSTRAINT chk_a_not_zero CHECK (a <> 0))
    ```
- Для нахождения каждого решения надо вычислять квадраты, дискриминанты
- С помощью производной таблицы это можно упростить:
  - ```
    SELECT
        IIF (D >= 0, (-b - sqrt(D)) / denom, NULL) AS sol_1,
        IIF (D > 0, (-b + sqrt(D)) / denom, NULL) AS sol_2
     FROM
        (SELECT b, b*b - 4*a*c, 2*a FROM coeffs) (b, D, denom)
    ```

Выборка из общих табличных выражений (CTE)

- CTE - Common Table Expressions
- Более сложный и мощный вариант производных таблиц
- Будем рассматривать отдельно позже
- Пример вычисления корней квадратного уравнения через CTE
  - ```
    WITH vars (b, D, denom) AS (
       SELECT b, b*b - 4*a*c, 2*a
       FROM coeffs
    )
    SELECT
       IIF (D >= 0, (-b - sqrt(D)) / denom, NULL) AS sol_1,
       IIF (D > 0, (-b + sqrt(D)) / denom, NULL) AS sol_2
    FROM vars
    ```
    ```

Оптимизация примера

```
WITH vars (b, D, denom) AS (
  SELECT b, b*b - 4*a*c, 2*a
  FROM coeffs),
 vars2 (b, D, denom, sqrtD) AS (
  SELECT
     b, D, denom,
     IIF (D >= 0, sqrt(D), NULL)
  FROM vars)
 SELECT
  IIF (D >= 0, (-b - sqrtD) / denom, NULL) AS sol_1,
  IIF (D > 0, (-b + sqrtD) / denom, NULL) AS sol_2
 FROM vars2
```

- Корень из дискриминанта вычисляется один раз, а не два

Соединение (JOIN)

- Соединяют данные двух источников в один набор
- Происходит для каждой строки и включает проверку условия соединения
- Результат также может быть соединен с другим набором другим соединением
- Существует несколько типов соединений со своими правилами
- Для примеров будем использовать две таблицы:
  - Таблица A

ID	S
87	Пушкин
35	Лермонтов

  - Таблица B

CODE	X
-23	56.77
87	416.0

Внутренние соединения (INNER JOIN)

- Соединяют два набора данных - левый и правый
- INNER JOIN включает только те строки, которые удовлетворяют условию соединения
- Например для запроса:
  - `SELECT * FROM` **`A JOIN B ON A.id = B.code`**
  - Будет следующий результат

ID	S	CODE	X
87	Пушкин	87	416.0

  - Только 1-я строка A соединена со 2-й строкой B.
  - Остальные строки не удовлетворяют условию соединения.
- Слово INNER является не обязательным

Внутренние соединения 2

- Возможны случаи когда строке левого набора соответствует несколько строк правого.
- Результат может быть примерно таким

ID	S	CODE	X
87	Пушкин	87	416.0
87	Пушкин	87	-1
-23	Есенин	-23	56.77
-23	Есенин	-23	34.55

Левое, правое и полное соединения

- Когда надо включить в результат все записи левого или правого набора
  - LEFT OUTER JOIN включает в результат все записи левого набора
  - RIGHT OUTER JOIN включает в результат все записи правого набора
  - FULL OUTER JOIN включает в результат все записи обоих наборов данных
- Если записи не нашлось пары по условию, значения заполняются NULL
- Ключевое слово OUTER является необязательным
- Например запрос
  - `SELECT * FROM A LEFT JOIN B ON A.id = B.code`
  - Вернет

ID	S	CODE	X
87	Пушкин	87	416.0
35	Лермонтов	<NULL>	<NULL>

Примеры внешних соединений

- Например запрос
  - `SELECT * FROM A RIGHT JOIN B ON A.id = B.code`
  - Вернет

ID	S	CODE	X
87	Пушкин	87	416.0
NULL	NULL	-23	56.77

- Например запрос
  - `SELECT * FROM A FULL JOIN B ON A.id = B.code`
  - Вернет

ID	S	CODE	X
87	Пушкин	87	416.0
NULL	NULL	-23	56.77
35	Лермонтов	NULL	NULL

Явные условия соединения

- Есть предложение **ON**
- Может содержать любое выражение проверки
- Обычно это:
  - Проверка на равенство
  - Ряд проверок с оператором AND
  - Называются **эквисоединениями**

Примеры с явным условием соединения

- ```
  SELECT * FROM customers c JOIN sales s ON s.cust_id = c.id
  WHERE c.city = 'Detroit'
  ```

- ```
  SELECT * FROM customers c LEFT JOIN sales s ON s.cust_id = c.id
  WHERE c.city = 'Detroit'
  ```

- ```
  SELECT m.fullname AS man, f.fullname AS woman
  FROM males m JOIN females f ON f.height > m.height
  ```

- ```
  SELECT p.firstname, p.middlename, p.lastname, c.name, m.name
  FROM pupils p JOIN classes c ON c.id = p.class
  LEFT JOIN mentors m ON m.id = p.mentor
  ```

Соединения с именованными столбцами

- Эквисоединения часто сравнивают столбцы с одинаковыми именами
- В таком случае можно использовать соединение с именованными столбцами
- Осуществляется с помощью USING
- Например,
  - `SELECT * FROM flotsam f JOIN jetsam j`
    `ON f.sea = j.sea AND f.ship = j.ship`
- Можно переписать так:
  - `SELECT * FROM flotsam JOIN jetsam` **`USING (sea, ship)`**
- Различия:
  - С явным условием каждый столбец будет включен в результат дважды.
  - С именованными столбцами только один раз. Очевидно у них одинаковые значения.

Естественные соединения (NATURAL JOIN)

- Основаны на соединениям с именованными столбцами
- Выполняют соединение по всем одноименным столбцам
- Типы столбцов должны совпадать
- Пусть даны две таблицы:
  - `CREATE TABLE TA (a BIGINT, s VARCHAR(12), ins_date DATE);`
  - `CREATE TABLE TB (a BIGINT, descr VARCHAR(12), x FLOAT, ins_date DATE);`
- Следующие два запроса эквивалентны:
  - `SELECT * FROM TA NATURAL JOIN TB;`
  - `SELECT * FROM TA JOIN TB USING (a, ins_date);`
- Аналогично другим соединениям можно добавить:
  - LEFT
  - RIGHT
  - FULL

Неявные соединения

- В стандарте SQL-89
  - Таблицы для соединения задаются указываются списком через запятую после FROM.
  - Условия задаются после WHERE.
- Такие соединения называются неявными
- Позволяет задавать только внутренние соединения
- Например,
  - ```
    SELECT * FROM customers c, sales s
    WHERE s.cust_id = c.id AND c.city = 'Detroit'
    ```
- Без условия после WHERE результатом будет декартово произведение
- В настоящее время не рекомендуется использовать
    ```

Перекрестное соединение (CROSS JOIN)

- Декартово произведение множеств записей из обеих частей
- Каждая строка левой части соединяется с каждой строкой правой
- Эквивалентно соединению по условию тавтологии (условие, которое всегда верно)
- Например, следующие два запроса эквивалентны:
  - `SELECT * FROM TA CROSS JOIN TB;`
  - `SELECT * FROM TA JOIN TB ON 1 = 1;`
- Полезны в случае когда надо получить все возможные комбинации из наборов значений.