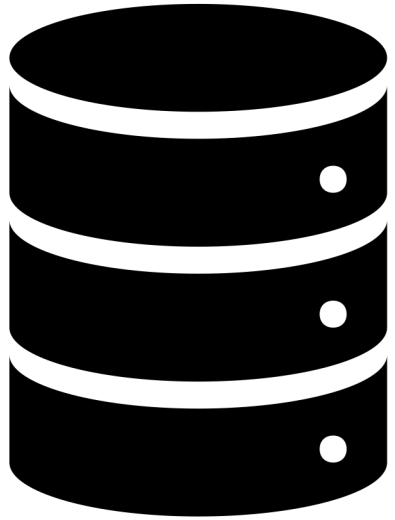
Базы данных



Лекция 6. Оконные функции.

Меркурьева Надежда
✉ merkurievanad@gmail.com
@merkurievanad

ФПМИ МФТИ, 2021

УЖЕ УМЕЕМ:

- Писать обычные запросы
- Писать запросы с использованием нескольких таблиц:
  - Различные виды JOIN’ов
  - Перечисление через запятую
- Оперировать с однотипными результатами запросов:
  - UNION (ALL)
  - INTERSECT
  - EXCEPT
- Использовать подзапросы

АНАЛИТИЧЕСКИЕ (ОКОННЫЕ) ФУНКЦИИ

- Принимают на вход столбец промежуточного результата вычисления и возвращают тоже столбец
- Местом их использования могут быть только разделы ORDER BY и SELECT
- Действуют подобно агрегирующим, но не уменьшают степень детализации
- Агрегируют данные порциями (окнами), количество и размер которых регулируется специальной синтаксической конструкцией

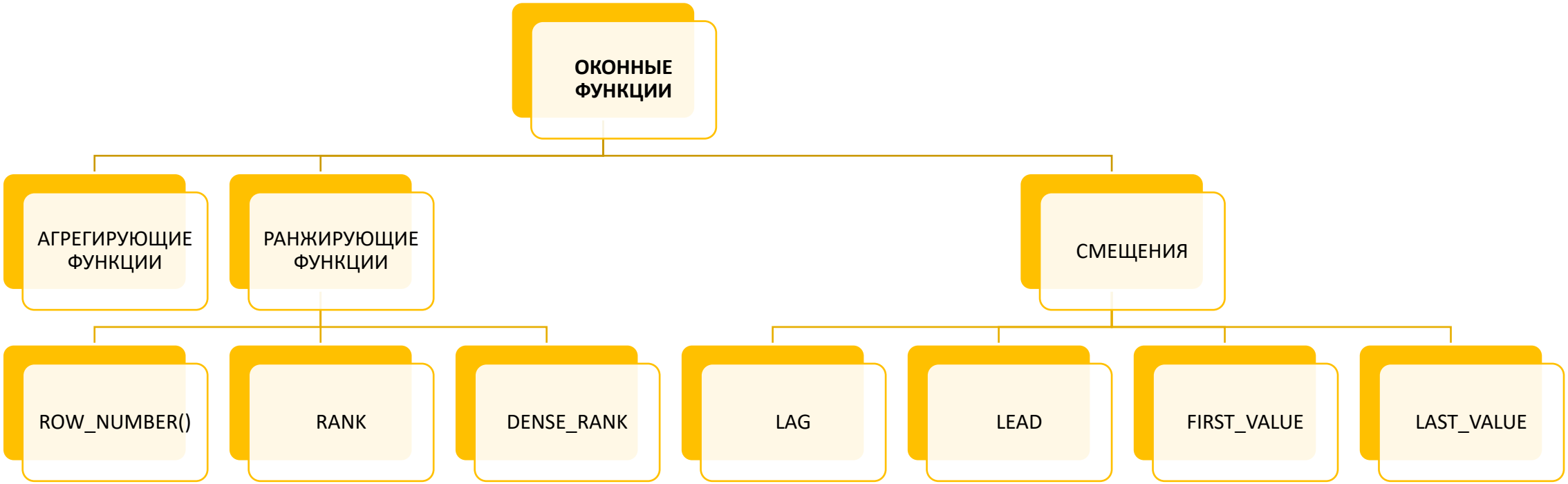
АНАЛИТИЧЕСКИЕ ФУНКЦИИ

```
an_function(expression) OVER (
   [PARTITION BY expression_comma_list_1]
   [ORDER BY expression_comma_list_2 [{ASC | DESC}]]
   [frame_clause])

frame_clause:
  {RANGE|ROWS} frame_start
  {RANGE|ROWS} BETWEEN frame_start AND frame_end

UNBOUNDED / offset PRECEDING
UNBOUNDED / offset FOLLOWING
CURRENT ROW
```

ОКОННЫЕ ФУНКЦИИ



ИСПОЛЬЗОВАНИЕ OVER

- `OVER` определяет «окно» или набор строк, которые будет использовать оконная функция, включая сортировку данных
- В выражении, которое задает оконную функцию, инструкция `OVER` ограничивает наборы строк с одинаковыми значениями в поле, по которому идет разделение
- Сама по себе инструкция `OVER()` не ограничена и содержит все строки из результирующего набора
- Инструкция `OVER` может многократно использоваться в одном `SELECT`, каждая со своим разделением и сортировкой

ИСПОЛЬЗОВАНИЕ OVER

```
OVER (
   [<PARTITION BY clause>]
   [<ORDER BY clause>]
   [<ROWS or RANGE clause>]
)
```

ПРАВИЛА СЕКЦИОНИРОВАНИЯ

- Внутри OVER необходимо указать поля таблицы, по которому будет скользить «окно», и правило, по которому строки будут секционироваться:
  - `PARTITION BY – отвечает за критерий секционирования`
  - `ORDER BY – отвечает за сортировку`
  - `ROWS | RANGE – дополнительные ограничения на диапазон строк окна (обязательно присутствие ORDER BY)`

PARTITION BY

- Логически разбивает множество на группы по критериям
- Аналитические функции применяются к группам независимо
- Если не указать конструкцию партиционирования, все множество считается одной группой

ORDER BY

- Задает критерий сортировки внутри каждой группы
- Агрегирующие функции чувствительны к наличию ORDER BY
- Если агрегирующая функция используется с ORDER BY, она становится «оконной»

PARTITION BY

```
SELECT territory_id
     , sales_amt
     , sum(sales_amt) OVER (PARTITION BY territory_id) as cumulative_sales_amt
     , sales_amt / sum(sales_amt) OVER (PARTITION BY territoty_id) * 100 as sales_total_prc
  FROM sales_person;
```

territory_id	sales_amt	cumulative_sales_amt	sales_total_prc	
NULL	559697,5639	1252127,9471	44,69	
NULL	172524,4512	1252127,9471	13,77	Окно 1
NULL	519905,9320	1252127,9471	41,52	
1	1573012,9383	4502152,2674	34,39	
1	1576562,1966	4502152,2674	35,01	Окно 2
1	1352577,1325	4502152,2674	30,04	
2	3763178,1787	3763178,1787	100,0	Окно 3
3	3189418,3662	3189418,3662	100,0	Окно 4
4	4251368,5497	6709904,1666	63,35	
4	2458535,6169	6709904,1666	36,65	Окно 5

ORDER BY

```
SELECT territory_id
     , sales_amt
     , sum(sales_amt) OVER (ORDER BY territore_id ASC) AS cumulative_sales_amt
  FROM sales_person;
```

territory_id	sales_amt	cumulative_sales_amt
NULL	559697,5639	1252127,9471
NULL	172524,4512	1252127,9471
NULL	519905,9320	1252127,9471
1	1573012,9383	5754280,2145
1	1576562,1966	5754280,2145
1	1352577,1325	5754280,2145
2	3763178,1787	9517458,3932
3	3189418,3662	12706876,7594
4	4251368,5497	19416780,9260
4	2458535,6169	19416780,9260

Нарастающий итог

АГРЕГИРУЮЩИЕ ОКОННЫЕ ФУНКЦИИ

```
SELECT business_entity, territory_id, year_num, sales_amt,
       sum(sales_amt) OVER (PARTITION BY territory_id) AS territory_ttl_amt,
       sum(sales_amt) OVER (PARTITION BY territory_id ORDER BY year_num) AS terr_year_ttl_amt
  FROM sales_person
 ORDER BY territory_id, year_num;
```

business_entity	territory_id	year_num	sales_amt	territory_ttl_amt	terr_year_ttl_amt
274	NULL	2005	559 697,5639	1 252 127,9471	559 697,5639
287	NULL	2006	172 524,4512	1 252 127,9471	732 222,0151
285	NULL	2007	519 905,932	1 252 127,9471	1 252 127,9471
283	1	2005	1 573 012,9383	4 502 152,2674	3 149 575,1349
280	1	2005	1 576 562,1966	4 502 152,2674	3 149 575,1349
284	1	2006	1 352 577,1325	4 502 152,2674	4 502 152,2674
275	2	2005	3 763 178,1787	3 763 178,1787	3 763 178,1787
277	3	2005	3 189 418,3662	3 189 418,3662	3 189 418,3662
276	4	2005	4 251 368,5497	6 709 904,1666	6 709 904,1666
281	4	2005	2 458 535,6169	6 709 904,1666	6 709 904,1666

ROWS

```
SELECT territory_id
     , sales_amt
     , sum(sales_amt) OVER (ORDER BY territory_id ASC ROWS BETWEEN CURRENT ROW AND 1
       FOLLOWING) AS cumulative_sales_amt
  FROM sales_person;
```

territory_id	sales_amt	cumulative_sales_amt	
NULL	559697,5639	732222,0151	Окно 1
NULL	172524,4512	692430,3832	
NULL	519905,9320	2092918,8703	
1	1573012,9383	3149575,1349	Окно 2
1	1576562,1966	2929139,3291	
1	1352577,1325	5115755,3112	
2	3763178,1787	6952596,5449	
3	3189418,3662	7440786,9159	
4	4251368,5497	6709904,1666	Окно N-1
4	2458535,6169	2458535,6169	Окно N

RANGE

```
SELECT territory_id
     , sales_amt
     , sum(sales_amt) OVER (ORDER BY territory_id ASC RANGE CURRENT ROW) AS cumulative_sales_amt
  FROM sales_person;
```

territory_id	sales_amt	cumulative_sales_amt	
NULL	172524,4512	1252127,9471	Окно 1
NULL	519905,9320	1252127,9471	
NULL	559697,5639	1252127,9471	
1	1352577,1325	4502152,2674	Окно 2
1	1573012,9383	4502152,2674	
1	1576562,1966	4502152,2674	
2	3763178,1787	3763178,1787	Окно 3
3	3189418,3662	3189418,3662	Окно 4
4	2458535,6169	6709904,1666	Окно 5
4	4251368,5497	6709904,1666	

ROWS

```
SELECT
  territory_id AS territory_id,
  business_entity AS business_entity,
  sales_amt AS sales_ytd,
  sum(sales_amt) OVER (ORDER BY territory_id ASC ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) AS cumulative_total_1,
  sum(sales_amt) OVER (ORDER BY territory_id ASC ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS cumulative_total_2,
  sum(sales_amt) OVER (PARTITION BY territory_id
                       ORDER BY business_entity ASC ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) AS cumulative_total_p1
FROM
  sales_person;
```

	territory_id	business_entity	sales_ytd	cumulative_total_1	cumulative_total_2	cumulative_total_p1
1	1	280	1576562.1966	5339740.3753	6912753.3136	3149575.1349
2	1	283	1573012.9383	3149575.1349	4502152.2674	2925590.0708
3	1	284	1352577.1325	2925590.0708	2925590.0708	1352577.1325
4	2	275	3763178.1787	6952596.5449	8529158.7415	3763178.1787
5	3	277	3189418.3662	5647953.9831	9411132.1618	3189418.3662
6	4	276	4251368.5497	4811066.1136	7269601.7305	6709904.1666
7	4	281	2458535.6169	6709904.1666	9899322.5328	2458535.6169
8	<null>	274	559697.5639	732222.0151	4983590.5648	1079603.4959
9	<null>	285	519905.9320	519905.932	692430.3832	692430.3832
10	<null>	287	172524.4512	692430.3832	1252127.9471	172524.4512

RANGE

```
SELECT
  territory_id AS terr_id,
  business_entity AS business_ent,
  sales_amt AS sales_ytd,
  sum(sales_amt) OVER (ORDER BY territory_id ASC RANGE CURRENT ROW) AS cml_ttl_cur,
  sum(sales_amt) OVER (ORDER BY territory_id ASC RANGE BETWEEN CURRENT ROW AND 1 FOLLOWING) AS cml_ttl_1,
  sum(sales_amt) OVER (ORDER BY territory_id ASC RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS cml_ttl_2,
  sum(sales_amt) OVER (PARTITION BY territory_id
                       ORDER BY business_entity ASC RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cml_ttl_p1,
  sum(sales_amt) OVER (PARTITION BY territory_id
                       ORDER BY business_entity ASC RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS cml_ttl_p2
FROM
  sales_person;
```

	terr_id	business_ent	sales_ytd	cml_ttl_cur	cml_ttl_1	cml_ttl_2	cml_ttl_p1	cml_ttl_p2
1	1	280	1576562.1966	4502152.2674	8265330.4461	8265330.4461	1576562.1966	4502152.2674
2	1	283	1573012.9383	4502152.2674	8265330.4461	8265330.4461	3149575.1349	2925590.0708
3	1	284	1352577.1325	4502152.2674	8265330.4461	8265330.4461	4502152.2674	1352577.1325
4	2	275	3763178.1787	3763178.1787	6952596.5449	11454748.8123	3763178.1787	3763178.1787
5	3	277	3189418.3662	3189418.3662	9899322.5328	13662500.7115	3189418.3662	3189418.3662
6	4	276	4251368.5497	6709904.1666	6709904.1666	9899322.5328	4251368.5497	6709904.1666
7	4	281	2458535.6169	6709904.1666	6709904.1666	9899322.5328	6709904.1666	2458535.6169
8	<null>	274	559697.5639	1252127.9471	1252127.9471	1252127.9471	559697.5639	1252127.9471
9	<null>	285	519905.9320	1252127.9471	1252127.9471	1252127.9471	1079603.4959	692430.3832
10	<null>	287	172524.4512	1252127.9471	1252127.9471	1252127.9471	1252127.9471	172524.4512

РАНЖИРУЮЩИЕ ОКОННЫЕ ФУНКЦИИ

- *row_number() – нумеруем каждую строку окна последовательно с шагом 1*
- *rank() – ранжируем каждую строку окна с разрывом в нумерации при равенстве значений*
- *dense_rank() – ранжируем каждую строку окна без разрывов в нумерации при равенстве значений*

РАНЖИРУЮЩИЕ ОКОННЫЕ ФУНКЦИИ

```
SELECT
  p.FirstName
  , p.LastName
  , a.PostalCode,
  , ROW_NUMBER()
      OVER (ORDER BY a.PostalCode) AS "Row Number"
  , RANK()
      OVER (ORDER BY a.PostalCode) AS "Rank"
  , DENSE_RANK()
      OVER (ORDER BY a.PostalCode) AS "Dense Rank"
FROM
  Person p
INNER JOIN
  Address a
ON
  a.AddressID = p.AddressID;
```

FirstName	LastName	PostalCode	Row Number	Rank	Dense Rank
Michael	Blythe	98027	1	1	1
Linda	Mitchell	98027	2	1	1
Jillian	Carson	98027	3	1	1
Garrett	Vargas	98027	4	1	1
Tsvi	Reiter	98027	5	1	1
Pamela	Ansman-Wolfe	98027	6	1	1
Shu	Ito	98055	7	7	2
José	Saraiva	98055	8	7	2
David	Campbell	98055	9	7	2
Tete	Mensa-Annan	98055	10	7	2
Lynn	Tsoflias	98055	11	7	2
Rachel	Valdez	98055	12	7	2
Jae	Pak	98055	13	7	2
Ranjit	Varkey Chudukatil	98055	14	7	2

Порядковый номер строки **Разрыв послед-ти** **Порядковый номер группы**

ФУНКЦИИ СМЕЩЕНИЯ

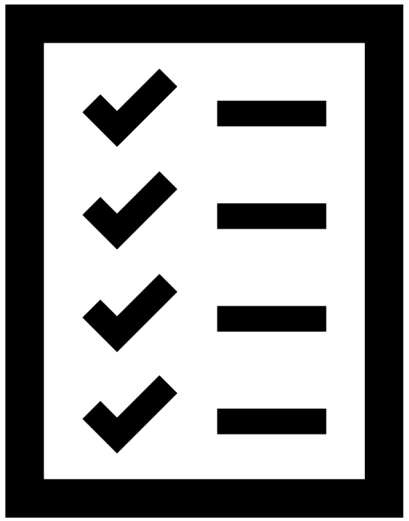
- *lag(attr, offset, default_value) – предыдущее значение со сдвигом*
- *lead(attr, offset, default_value) – следующее значение со сдвигом*
- *first_value(attr) – первое значение в окне с первой по текущую строку*
- *last_value(attr) – последнее значение в окне с первой по текущую строку*

ФУНКЦИИ СМЕЩЕНИЯ

```
SELECT business_entity_id, year_num, current_quota_amt
     , LAG(current_quota_amt, 1, 0) OVER (ORDER BY year_num) AS prev_quota_amt
     , LEAD(current_quota_amt, 1, 0) OVER (ORDER BY year_num) AS next_quota_amt
  FROM sales_person_quota_history
 WHERE business_entity_id = 275;
```

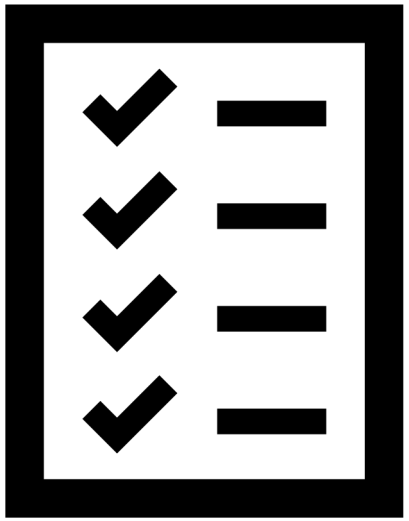
business_entity_id	year_num	current_quota_amt	prev_quota_amt	next_quota_amt
275	2005	367000	0	556000
275	2005	556000	367000	502000
275	2006	502000	556000	550000
275	2006	550000	502000	1429000
275	2006	1429000	550000	1324000
275	2006	1324000	1429000	0

ТИПОВЫЕ ЗАДАЧИ SQL



- Поддержка версионности таблиц
- Соединение версионных таблицы
- Поиск разрывов в версионности
- Поиск пересечений в версионности
- Сравнение таблиц

ТИПОВЫЕ ЗАДАЧИ SQL



- Поддержка версионности таблиц
- Соединение версионных таблицы

- Поиск разрывов в версионности
- Поиск пересечений в версионности
- Сравнение таблиц

ПОИСК РАЗРЫВОВ И ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

- Версионные таблицы хранят историю изменения атрибутов
- История изменения атрибутов представляет собой набор временных интервалов
- Зачастую требуется проверка истории на наличие разрывов или пересечений в версионности

ПОИСК РАЗРЫВОВ В ВЕРСИОННОСТИ

EMPLOYEE_NM	DEPT_NO	VALID_FROM_DTTM	VALID_TO_DTTM
SMITH	1	1995-05-01 00:00:00	2000-06-15 23:59:59
SMITH	2	2000-06-16 00:00:00	5999-02-01 00:00:00
JONES	1	1996-05-10 00:00:00	1997-11-10 23:59:59
JONES	2	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1	1996-07-20 00:00:00	1997-01-15 23:59:59
BLAKE	2	1997-01-16 00:00:00	2002-05-15 23:59:59
BLAKE	3	2002-05-18 00:00:00	5999-01-01 00:00:00

ПОИСК РАЗРЫВОВ В ВЕРСИОННОСТИ

EMPLOYEE_NM	DEPT_NO	VALID_FROM_DTTM	VALID_TO_DTTM
SMITH	1	1995-05-01 00:00:00	2000-06-15 23:59:59
SMITH	2	2000-06-16 00:00:00	5999-02-01 00:00:00
JONES	1	1996-05-10 00:00:00	1997-11-10 23:59:59
JONES	2	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1	1996-07-20 00:00:00	1997-01-15 23:59:59
BLAKE	2	1997-01-16 00:00:00	2002-05-15 23:59:59
BLAKE	3	2002-05-18 00:00:00	5999-01-01 00:00:00

В таблице присутствуют 2 разрыва:

1. Для сотрудника ‘JONES’ нет информации о его работе в период с ‘1997-11-11’ по ‘1999-03-04’
2. Для сотрудника ‘BLAKE’ нет информации о его работе в период с ‘2002-05-16’ по ‘2002-05-17’

РАЗРЫВЫ В ВЕРСИОННОСТИ

- Логика исходных данных
  - сотрудник ‘JONES’ не работал в компании с ‘1997-11-11’ по ‘1999-03-04’
- Ошибка ввода данных
  - у сотрудника ‘BLAKE’ неправильно введена дата начала его работы в отделе 3 или дата окончания его работы в отделе 2

ПОИСК РАЗРЫВОВ В ВЕРСИОННОСТИ

- Использование аналитических функций
- Разбить данные на группы по ключу таблицы:
  - по `EMPLOYEE_NM`
- Отсортировать данные в группах по дате начала действия интервала:
  - `VALID_FROM_DTTM`
- Проверить, превышает ли разница между датой начала действия текущего интервала и датой окончания действия предыдущего интервал в 1 секунду

ПОИСК РАЗРЫВОВ В ВЕРСИОННОСТИ

```
select employee_nm
     , prev_valid_to_dttm
     , valid_from_dttm
  from (
        select employee_nm
             , lag(valid_to_dttm)
                 over (partition by employee_nm
                                 order by valid_from_dttm) as prev_valid_to_dttm
             , valid_from_dttm
          from emp_dept
       ) t
 where valid_from_dttm - prev_valid_to_dttm > interval '1 sec';
```

EMPLOYEE_NM	PREV_VALID_TO_DTTM	VALID_FROM_DTTM
JONES	1997-11-10 23:59:59	1999-03-05 00:00:00
BLAKE	2002-05-15 23:59:59	2002-05-18 00:00:00

ПОИСК РАЗРЫВОВ В ВЕРСИОННОСТИ

```
select employee_nm
     , prev_valid_to_dttm + interval '1 sec' as miss_valid_from_dttm
     , valid_from_dttm - interval '1 sec'    as miss_valid_to_dttm
  from (
        select employee_nm
             , lag(valid_to_dttm)
                 over (partition by employee_nm
                                order by valid_from_dttm) as prev_valid_to_dttm
             , valid_from_dttm
          from emp_dept
       ) t
 where valid_from_dttm - prev_valid_to_dttm > interval '1 sec';
```

EMPLOYEE_NM	MISS_VALID_FROM_DTTM	MISS_VALID_TO_DTTM
JONES	1997-11-11 00:00:00	1999-03-04 23:59:59
BLAKE	2002-05-16 00:00:00	2002-05-17 23:59:59

ПОИСК ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

EMPLOYEE_NM	DEPT_NO	VALID_FROM_DTTM	VALID_TO_DTTM
SMITH	1	1995-05-01 00:00:00	2000-06-15 23:59:59
SMITH	2	2000-06-16 00:00:00	5999-01-01 00:00:00
JONES	1	1996-05-10 00:00:00	1999-07-20 23:59:59
JONES	2	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1	1996-07-20 00:00:00	1997-01-15 23:59:59
BLAKE	2	1997-01-16 00:00:00	2002-05-20 23:59:59
BLAKE	3	2002-05-18 00:00:00	5999-01-01 00:00:00

ПОИСК ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

EMPLOYEE_NM	DEPT_NO	VALID_FROM_DTTM	VALID_TO_DTTM
SMITH	1	1995-05-01 00:00:00	2000-06-15 23:59:59
SMITH	2	2000-06-16 00:00:00	5999-01-01 00:00:00
JONES	1	1996-05-10 00:00:00	1999-07-20 23:59:59
JONES	2	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1	1996-07-20 00:00:00	1997-01-15 23:59:59
BLAKE	2	1997-01-16 00:00:00	2002-05-20 23:59:59
BLAKE	3	2002-05-18 00:00:00	5999-01-01 00:00:00

В таблице присутствуют 2 пересечения:

1. Сотрудник ‘JONES’ работал в двух отделах в период с ‘1999-03-05’ по ‘1999-07-20’
2. Сотрудник ‘BLAKE’ работал в двух отделах в период с ‘2002-05-18’ по ‘2002-05-20’

ПЕРЕСЕЧЕНИЯ В ВЕРСИОННОСТИ

- Логика исходных данных
  - сотрудник ‘JONES’ участвовал в проектах двух разных отделов с ‘1999-03-05’ по ‘1999-07-20’
- Ошибка ввода данных
  - у сотрудника ‘BLAKE’ неправильно введена дата начала его работы в отделе 3 или дата окончания его работы в отделе 2

ПОИСК ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

- Использование аналитических функций
- Разбить данные на группы по ключу таблицы:
  - `EMPLOYEE_NM`
- Отсортировать данные в группах по дате начала действия интервала:
  - `VALID_FROM_DTTM`
- Проверить, не начинается ли период действия текущей записи раньше, чем заканчивается предыдущий интервал

Поиск пересечений в версионности

```
select employee_nm
     , prev_valid_from_dttm
     , prev_valid_to_dttm
     , valid_from_dttm
     , valid_to_dttm
  from (
        select employee_nm
             , lag(valid_from_dttm)
                 over (partition by employee_nm
                           order by valid_from_dttm) as prev_valid_from_dttm,
             , lag(valid_to_dttm)
                 over (partition by employee_nm
                           order by valid_from_dttm) as prev_valid_to_dttm,
             , valid_from_dttm
             , valid_to_dttm
          from emp_dept
       ) t
 where valid_from_dttm <= prev_valid_to_dttm;
```

EMPLOYEE_NM	PREV_VALID_FROM_DTTM	PREV_VALID_TO_DTTM	VALID_FROM_DTTM	VALID_TO_DTTM
JONES	1996-05-10 00:00:00	1999-07-20 23:59:59	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1997-01-16 00:00:00	2002-05-20 23:59:59	2002-05-18 00:00:00	5999-01-01 00:00:00

ПОИСК ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

- Можно решить эту задачу без аналитический функций простым соединением версионных таблиц
- Условия использования такого решения:
  - В качестве соединяемых таблиц будет выступать одна и та же таблица
  - Наличие возможности отличить любые две строки таблицы
  - Нет строк с одинаковым ключом и периодом действия

ПОИСК ПЕРЕСЕЧЕНИЙ В ВЕРСИОННОСТИ

```
select e1.employee_name
     , e1. as prev_valid_from_dttm
     , e1.valid_to_dttm as prev_valid_to_dttm
     , e2.valid_from_dttm
     , e2.valid_to_dttm
  from emp_dept e1
 inner join emp_dept e2
    on e1.employee_nm = e2.employee_nm
   and e2.valid_from_dttm between e1.valid_from_dttm and e1.valid_to_dttm
   and e1.valid_to_dttm <> e2.valid_to_dttm;
```

EMPLOYEE_NM	PREV_VALID_FROM_DTTM	PREV_VALID_TO_DTTM	VALID_FROM_DTTM	VALID_TO_DTTM
JONES	1996-05-10 00:00:00	1999-07-20 23:59:59	1999-03-05 00:00:00	2001-04-25 23:59:59
BLAKE	1997-01-16 00:00:00	2002-05-20 23:59:59	2002-05-18 00:00:00	5999-01-01 00:00:00

СРАВНЕНИЕ ТАБЛИЦ

- Нередко возникает необходимость сравнения двух вариантов одной и той же таблицы и построения ***таблицы с различиями (diff-таблицы)***

СРАВНЕНИЕ ТАБЛИЦ

EMP

employee_nm	job_title_nm	hire_dt	salary_amt
ADAMS	CLERK	1987-06-23	1100
ALLEN	SALESMAN	1981-02-20	2000
BLAKE	MANAGER	1981-05-01	2850
CLARK	MANAGER	1981-06-09	2450
JAMES	CLERK	1981-12-03	950
JONES	MANAGER	1981-04-02	2975

EMP_OLD

employee_nm	job_title_nm	hire_dt	salary_amt
ADAMS	CLERK	1987-05-23	1100
ALLEN	SALESMAN	1981-02-20	1600
BLAKE	MANAGER	1981-05-01	2850
FORD	ANALYST	1981-12-03	3000
JAMES	CLERK	1981-12-03	950

Хотим сравнить таблицы EMP_OLD (до внесения изменений пользователей), EMP – после.
Были внесены следующие изменения:

- Добавлены сотрудники ‘CLARK’ и ‘JONES’ (операции INSERT)
- Удален сотрудник ‘FORD’ (операция DELETE)
- Изменена дата рождения сотрудника ‘ADAMS’ и зарплата сотрудника ‘ALLEN’ (операции UPDATE).

СРАВНЕНИЕ ТАБЛИЦ

- Таблица строк `EMP`, отсутствующих в `EMP_OLD`
  - результаты выполнения операций `INSERT` и `UPDATE`
- Таблица изменений неключевых полей
  - результаты работы операций `UPDATE`
- Таблица изменений в ключевых полях
  - результаты работы операций `INSERT` и `DELETE`
- Таблица всех изменений

СРАВНЕНИЕ ТАБЛИЦ

- Таблица строк EMP, отсутствующих в EMP_OLD

```
SELECT employee_nm
     , job_title_nm
     , hire_dt
     , salary_amt
  FROM emp
EXCEPT
SELECT employee_nm
     , job_title_nm
     , hire_dt
     , salary_amt
  FROM emp_old;
```

employee_nm	job_title_nm	hire_dt	salary_amt
ADAMS	CLERK	1987-06-23	1100
ALLEN	SALESMAN	1981-02-20	2000
CLARK	MANAGER	1981-06-09	2450
JONES	MANAGER	1981-04-02	2975

СРАВНЕНИЕ ТАБЛИЦ

- Таблица изменений неключевых полей

```
SELECT emp.employee_nm,
     , emp_old.job_title_nm AS old_job_title_nm
     , emp.job_title_nm     AS new_job_title_nm
     , emp_old.hire_dt      AS old_hire_dt
     , emp.hire_dt          AS new_hire_dt
     , emp_old.salary_amt   AS old_salary_amt
     , emp.salary_amt       AS new_salary_amt
  FROM emp
 INNER JOIN emp_old
    ON emp.employee_nm = emp_old.employee_nm
   AND (emp.job_title_nm IS DISTINCT FROM emp_old.job_title_nm
        OR emp.hire_dt IS DISTINCT FROM emp_old.hire_dt
        OR emp.salary_amt IS DISTINCT FROM emp_old.salary_amt);
```

employee_nm	old_job_title_nm	new_job_title_nm	old_hire_dt	new_hire_dt	old_salary_amt	new_salary_amt
ADAMS	CLERK	CLERK	1987-05-23	1987-06-23	1100	1100
ALLEN	SALESMAN	SALESMAN	1981-02-20	1981-02-20	1600	2000

СРАВНЕНИЕ ТАБЛИЦ

- Таблица изменений в ключевых полях

```
SELECT coalesce(emp.employee_nm, emp_old.employee_nm)    AS employee_nm
     , coalesce(emp.job_title_nm, emp_old.job_title_nm) AS job_title_nm
     , coalesce(emp.hire_dt, emp_old.hire_dt)           AS hire_dt
     , coalesce(emp.salary_amt, emp_old.salary_amt)     AS salary_amt
     , CASE
         WHEN emp.employee_nm IS NULL
           THEN 'DELETE'
         ELSE 'INSERT'
       END AS dml_operation
  FROM emp
  FULL JOIN emp_old
    ON emp.employee_nm = emp_old.employee_nm
 WHERE emp.employee_nm IS NULL
    OR emp_old.employee_nm IS NULL;
```

employee_nm	job_title_nm	hire_dt	salary_amt	dml_operation
CLARK	MANAGER	1981-06-09	2450	INSERT
JONES	MANAGER	1981-04-02	2975	INSERT
FORD	ANALYST	1981-12-03	3000	DELETE

СРАВНЕНИЕ ТАБЛИЦ: ВСЕ ИЗМЕНЕНИЯ

```
SELECT CASE
         WHEN emp.employee_nm IS NULL
           THEN 'DELETE'
         WHEN emp_old.employee_nm IS NULL
           THEN 'INSERT'
         ELSE 'UPDATE'
       END                                                    AS dml_operation
     , coalesce(emp.employee_nm, emp_old.employee_nm)         AS employee_nm
     , emp_old.job_title_nm                                   AS old_job_title_nm
     , emp.job_title_nm                                       AS new_job_title_nm
     , emp_old.hire_dt                                        AS old_hire_dt
     , emp.hire_dt                                            AS new_hire_dt
     , emp_old.salary_amt                                     AS old_salary_amt
     , emp.salary_amt                                         AS new_salary_amt
  FROM emp
  FULL JOIN emp_old
    ON emp.employee_nm = emp_old.employee_nm
 WHERE emp.employee_nm IS NULL
    OR emp_old.employee_nm IS NULL
    OR emp.job_title_nm IS DISTINCT FROM emp_old.job_title_nm
    OR emp.hire_dt IS DISTINCT FROM emp_old.hire_dt
    OR emp.salary_amt IS DISTINCT FROM emp_old.salary_amt;
```

СРАВНЕНИЕ ТАБЛИЦ: ВСЕ ИЗМЕНЕНИЯ

dml_operation	employee_nm	old_job_title_nm	new_job_title_nm	old_hire_dt	new_hire_dt	old_salary_amt	new_salary_amt
UPDATE	ADAMS	CLERK	CLERK	1987-05-23	1987-06-23	1100	1100
UPDATE	ALLEN	SALESMAN	SALESMAN	1981-02-20	1981-02-20	1600	2000
INSERT	CLARK	NULL	MANAGER	NULL	1981-06-09	NULL	2450
INSERT	JONES	NULL	MANAGER	NULL	1981-04-02	NULL	2975
DELETE	FORD	ANALYST	NULL	1981-12-03	NULL	3000	NULL