??????????, ??????????? ? ???????????

Логическое объединение таблиц

объединяются по толбцам последовательно. Лучше одинаковые псевдонимы.
union (или) объединяет построчно +сортировка +удаление дубликатов, union all без +. Есть предопрделенный необязательный столбец typ для ссылки на источник для строки.
intersect (И)
except (не) все что есть в первом исключая повторяющиеся со вторым
order by можно добавить после последнего запроса, относится к первому
выполнение сверху вниз, но intersect высший приоритет, можно использовать скобки

Соединения таблиц

ограничение внешнего ключа не является обязательным условием, чтобы соединить две таблицы. on - условие соединения таблиц. Если во всех таблицах совпадают названия столбцов, то USING (address_id); Using лучше не использовать.

```
SELECT c.first_name, c.last_name, time(r.rental_date) rental_time
FROM customer c
INNER JOIN rental r
ON c.customer_id = r.customer id WHERE date(r.rental_date) = '2015-04-03';
```

несколько таблиц, порядок перечисления значения не имеет:

```
select c.first_name, c.last_name, ci.city
from city ci
inner join address a on a.address_id = ci.address_id
inner join customer c on a.address_id = c.address_id;
```

с подзапросом:

```
SELECT c.first_name, c.last_name, addr.address, addr.city
FROM customer c
INNER JOIN
  (SELECT a.address_id, a.address, ct.city
   FROM address a
   INNER JOIN city ct ON a.city_id = ct.city_id WHERE a.district = 'California') addr
```

```
ON c.address id = addr.address id;
```

если в inner join несколько раз соединение с одной таблицей, то псевдонимы обязательны. возможно ссылаться на саму себя (например если в таблице есть строка, указывающая на родителя)

Типы соединений

- inner join - на каждую строку из левой таблицы строки из правой. (оператор И)
- left/right outer join - все строки левой таблицы, если в правой есть то добавляем иначе null
- cross join - перекрестное (чистое декартовое) соединение. Используется при создании новых данных.
- natural join - типа естественное соединение, предоставляющее выбор типа базе. Лучше не использовать.

Пример cross join

```
SELECT days.dt, COUNT(r.rental_id) num_rentals FROM rental r
RIGHT OUTER JOIN (SELECT DATE_ADD('2005-01-01', INTERVAL (ones.num + tens.num + hundreds.num)
DAY) dt
FROM
(SELECT 0 num UNION ALL
SELECT 1 num UNION ALL
SELECT 2 num UNION ALL
SELECT 3 num UNION ALL
SELECT 4 num UNION ALL
SELECT 5 num UNION ALL
SELECT 6 num UNION ALL
SELECT 7 num UNION ALL
SELECT 8 num UNION ALL
SELECT 9 num) ones
CROSS JOIN
(SELECT 0 num UNION ALL
SELECT 10 num UNION ALL
SELECT 20 num UNION ALL '
SELECT 30 num UNION ALL
SELECT 40 num UNION ALL
SELECT 50 num UNION ALL
SELECT 60 num UNION ALL
SELECT 70 num UNION ALL
SELECT 80 num UNION ALL
```

```
SELECT 90 num) tens
CROSS JOIN
(SELECT 0 num UNION ALL
SELECT 100 num UNION ALL
SELECT 200 num UNION ALL
SELECT 300 num) hundreds
WHERE DATE_ADD('2005-01-011, INTERVAL (ones.num + tens.num + hundreds.num) DAY) < '2006-01-01'
days
ON days.dt = date(r.rental_date) GROUP BY days.dt ORDER BY 1
```

Группировка

```
SELECT customer_id, count(*) FROM rental GROUP BY customer id;
```

сортировка по count - либо по номеру, либо определив имя столбца и по имени. Условия для агрегатных функций до создания полного набора, поэтому where count(*) > 40 нельзя. Можно having SELECT customer_id, count(*) FROM rental GROUP BY customer_id HAVING count(*) >= 40; но where можно для условий данных WHERE f.rating IN ('G', 'PG')

Агрегатные функции:
count(*)

```
COUNT(DISTINCT customer_id)
```

```
SELECT
  MAX(amount) max_amt,
  MIN(amount) min_amt,
  AVG(amount) avg_amt,
  SUM(amount) tot_amt,
  COUNT(*) num_payments
FROM payment;
```

агрегаты применяются либо ко всей выборке, либо к группам определенным в group by
в аргументе агрегатов любые функции, возвр число строку дату
null: игнорируется при рассчетах. Но count(*) отдает кол-во строк, и null считается. Т е count(val) число значений столбца без null, count(*) число строк в столбце val

Способы группировки

Множественная:

```
SELECT fa.actor_id, f.rating, count(*)
FROM film_actor fa INNER JOIN film f ON fa.film_id = f.film_id
GROUP BY fa.actor_id, f.rating ORDER BY 1,2;
```

Выражения:

```
SELECT extract(YEAR FROM rental_date) year, COUNT(*) how_many
FROM rental
GROUP BY extract(YEAR FROM rental date);
```

Добавка подсумм:

```
GROUP BY fa.actor_id, f.rating WITH ROLLUP
```

Непересекающееся множество DISTINCT (затратная операция)

```
SELECT DISTINCT actor_id FROM film_actor ORDER BY actor_id;
```

В столбцы можно добавлять вычисляемые поля
order by несколько столбцов
desc по убыванию ORDER BY time(r.rental date) desc;
можно использовать номер столбца в перечислении

Revision #2
Created 7 April 2025 05:10:26 by Admin
Updated 7 April 2025 05:35:52 by Admin