ДИСЦИПЛИНА

«Базы данных (PostgreSQL)»

1 Описание задания

1.1 Общая информация

Задание на финал состоит из двух частей: проектирование БД на основе существующих данных и проектирование представлений (views) для их анализа.

Для выполнения задания участнику предоставляется дамп базы данных с одной таблицей public.input_data.

Поле data указанной таблицы содержит информацию в формате jsonb. Подробный формат данных представлен в разделе 2.

Предметная область - система автоматического сбора показаний приборов учёта. В данных можно встретить такие сущности как дома, помещения, индивидуальные и общедомовые приборы учёта, история их показаний и поверок.

Результатом работы участника является единый SQL-файл, который при запуске создаст хранилище и представления.

1.2 Требования к проектируемой БД

Необходимо создать схему vit. Вся работа проводится в данной схеме. Вносить изменения в таблицу public.input_data запрещено.

Участник сам определяет набор сущностей, связей, индексов, а также используемые типы данных.

Для заполнения созданной схемы данными участник должен создать процедуру vit.fill_data(). Данная процедура при запуске должна заполнить созданные таблицы данными из public.input_data.

Результирующий SQL-файл НЕ ДОЛЖЕН содержать вызов процедуры vit.fill_data(), только её создание.

1.3 Создание представлений

На этом этапе работа также проводится исключительно в схеме vit. Использовать в представлениях таблицу public.input_data запрещено.

Содержание каждого представления описано в соответствующем разделе. При проверке выборка будет производиться с определёнными фильтрами. Для каждого представления будут указаны используемые для фильтрации поля. Поиск по текстовым полям будет выполняться в формате lower(field) LIKE '%search%', если не указано иное.

Пустые числовые значения (например, при агрегации) заменять не нужно, оставить NULL.

1.3.1 vit.values_without_verification

Необходимо вывести информацию о показаниях, снятых вне срока действия поверки прибора.

Поля представления:

- street - название улицы;
- house - номер дома;
- manufacturer - производитель ПУ;
- model - модель ПУ;
- serial_number - серийный номер;
- date - дата и время снятия показаний;
- t1 - показание по тарифу 1;
- t2 - показание по тарифу 2;
- t3 - показание по тарифу 3.

Фильтрация будет проводиться по полям street и house.

Сортировка в следующем порядке street, house, manufacturer, model, serial_number, date.

1.3.2 vit.values_bihourly

Необходимо вывести показания общедомовых приборов учёта, разбитые по сетке с частотой 2 часа. Т.е. последнее показание на 00:00, на 02:00, на 04:00 и т.д.

Поля представления:

- street - название улицы;
- house - номер дома;
- manufacturer - производитель ПУ;
- model - модель ПУ;
- serial_number - серийный номер;
- date - дата и время по сетке;
- t1 - показание по тарифу 1;
- t2 - показание по тарифу 2;
- t3 - показание по тарифу 3.

Фильтрация будет проводиться по полям street, house, serial_number, date. Фильтрация по serial_number ведётся по точному совпадению.

Сортировка в следующем порядке street, house, manufacturer, model, serial_number, date.

1.3.3 vit.individual_consumption_by_resource

Необходимо вывести потребление ресурсов по-суточно по индивидуальным приборам учёта.

Поля представления:

- street - название улицы;
- house - номер дома;
- resource - ресурс (числом);
- date - дата;
- t1 - потребление по тарифу 1;
- t2 - потребление по тарифу 2;
- t3 - потребление по тарифу 3.

Фильтрация будет проводиться по полям street, house, date.

Сортировка в следующем порядке street, house, resource, date.

1.3.4 vit.meter_models

Необходимо вывести количество установленных в настоящее время приборов учёта с разбивкой по моделям.

Поля представления:

- manufacturer - производитель ПУ;
- model - модель ПУ;
- count - количество приборов.

Фильтрация проводиться не будет.

Сортировка в следующем порядке manufacturer, model.

1.3.5 vit.monthly_consumption

Необходимо вывести потребление по дому по месяцам в разрезе ресурсов через ИПУ, и через ОДПУ.

Поля представления:

- street - название улицы;
- house - номер дома;
- year - год;
- month - месяц (числом);
- resource - ресурс (числом);
- im_t1 - потребление по тарифу 1, рассчитанное по ИПУ;
- im_t2 - потребление по тарифу 2, рассчитанное по ИПУ;
- im_t3 - потребление по тарифу 3, рассчитанное по ИПУ.
- hm_t1 - потребление по тарифу 1, рассчитанное по ОДПУ;
- hm_t2 - потребление по тарифу 2, рассчитанное по ОДПУ;
- hm_t3 - потребление по тарифу 3, рассчитанное по ОДПУ.

Фильтрация будет проводиться по полям street, house, year, month.

Сортировка в следующем порядке street, house, year, month, resource.

2 Формат входных данных

Описание модели с комментариями на языке C#:

```
/// <summary>
/// Входные данные.
/// </summary>
[Comment("Входные данные.")]
[Table("input_data", Schema = "public")]
public class InputDatum
{
    /// <summary>
    /// Дом.
    /// </summary>
    public class House
    {
        /// <summary>
        /// Прибор учёта.
        /// </summary>
        public class Meter
        {
            /// <summary>
            /// Измеряемый ресурс.
            /// </summary>
            public enum ResourceType
            {
                /// <summary>
                /// Холодная вода.
                /// </summary>
                ColdWater = 0,

                /// <summary>
                /// Горячая вода.
                /// </summary>
                HotWater = 1,

                /// <summary>
                /// Электроэнергия.
                /// </summary>
                Electricity = 2,

                /// <summary>
                /// Природный газ.
                /// </summary>
                Gas = 3,
            }

            /// <summary>
            /// Показание прибора.
            /// </summary>
            public class Value
            {
                /// <summary>
```

```
    /// Дата и время снятия показания.
    /// </summary>
    [JsonPropertyName("d")]
    public DateTimeOffset Date { get; set; }

    /// <summary>
    /// Показание по первому тарифу.
    /// </summary>
    [JsonPropertyName("t1")]
    public decimal FirstTariffValue { get; set; }

    /// <summary>
    /// Показание по второму тарифу.
    /// </summary>
    [JsonPropertyName("t2")]
    public decimal? SecondTariffValue { get; set; }

    /// <summary>
    /// Показание по третьему тарифу.
    /// </summary>
    [JsonPropertyName("t3")]
    public decimal? ThirdTariffValue { get; set; }
}

/// <summary>
/// Информация о поверке.
/// </summary>
public class Verification
{
    /// <summary>
    /// Дата поверки.
    /// </summary>
    [JsonPropertyName("d")]
    public DateOnly Date { get; set; }

    /// <summary>
    /// Действительна до.
    /// </summary>
    [JsonPropertyName("to")]
    public DateOnly ValidTo { get; set; }
}

/// <summary>
/// Измеряемый ресурс.
/// </summary>
[JsonPropertyName("res")]
public ResourceType Resource { get; set; }

/// <summary>
/// Производитель.
/// </summary>
[JsonPropertyName("man")]
public string Manufacturer { get; set; } = string.Empty;

/// <summary>
```

```
/// Модель.
/// </summary>
[JsonPropertyName("mod")]
public string Model { get; set; } = string.Empty;

/// <summary>
/// Серийный номер.
/// </summary>
[JsonPropertyName("sn")]
public string SerialNumber { get; set; } = string.Empty;

/// <summary>
/// Межповерочный интервал, лет.
/// </summary>
[JsonPropertyName("vi")]
public int VerificationInterval { get; set; }

/// <summary>
/// Дата установки.
/// </summary>
[JsonPropertyName("id")]
public DateOnly InstallationDate { get; set; }

/// <summary>
/// Дата демонтажа.
/// </summary>
[JsonPropertyName("dd")]
public DateOnly? DismantlingDate { get; set; }

/// <summary>
/// Используется первый тариф.
/// </summary>
[JsonPropertyName("t1")]
public bool IsFirstTariffInUse { get; set; } = true;

/// <summary>
/// Используется второй тариф.
/// </summary>
[JsonPropertyName("t2")]
public bool IsSecondTariffInUse { get; set; }

/// <summary>
/// Используется третий тариф.
/// </summary>
[JsonPropertyName("t3")]
public bool IsThirdTariffInUse { get; set; }

/// <summary>
/// История показаний.
/// </summary>
[JsonPropertyName("v")]
public List<Value> Values { get; set; } = [];

/// <summary>
/// История поверок.
```

```
        /// </summary>
        [JsonPropertyName("vh")]
        public List<Verification> VerificationHistory { get; set; } = [];
    }

    /// <summary>
    /// Помещение.
    /// </summary>
    public class Apartment
    {
        /// <summary>
        /// Номер помещения.
        /// </summary>
        [JsonPropertyName("n")]
        public string Number { get; set; } = string.Empty;

        /// <summary>
        /// Приборы учёта в помещении.
        /// </summary>
        [JsonPropertyName("m")]
        public List<Meter> ApartmentMeters { get; set; } = [];
    }

    /// <summary>
    /// Улица.
    /// </summary>
    [JsonPropertyName("s")]
    public string Street { get; set; } = string.Empty;

    /// <summary>
    /// Номер дома.
    /// </summary>
    [JsonPropertyName("n")]
    public string Number { get; set; } = string.Empty;

    /// <summary>
    /// Помещения в доме.
    /// </summary>
    [JsonPropertyName("a")]
    public List<Apartment>? Apartments { get; set; }

    /// <summary>
    /// Общедомовые приборы учёта.
    /// </summary>
    [JsonPropertyName("hm")]
    public List<Meter> HouseMeters { get; set; } = [];
}

/// <summary>
/// Идентификатор записи.
/// </summary>
[Column("id")]
[Comment("Идентификатор записи.")]
[Key]
public long Id { get; set; }
```

```
    /// <summary>
    /// Данные записи.
    /// </summary>
    [Column("data")]
    [Comment("Данные записи.")]
    [Required]
    public House Data { get; set; } = new();
}
```

3 Оценка результатов

Распределение баллов:

№	Категория	Максимальное количество баллов
1	Перенос данных	25
2	Архитектура БД	25
3	Представления на БД малого размера (N штук)	25
4	Представления на БД большого размера (N штук)	25
Итого		100

Работа участника не будет оцениваться, если единый SQL-скрипт не выполняется. Перед сдачей работы протестируйте скрипт. Например, можете изменить название схемы на vit2 и выполнить. Но не отправьте случайно вариант с неправильной схемой на проверку, допускается использование только vit.

До 25 баллов может получить участник за перенос данных. Оценивается как полнота переноса сущностей предметной области, так и полнота переноса самих данных. Учтите, если процедура vit.fill_data() не выполняется за конечное время на БД большого размера, то результат аннулируется.

До 25 баллов может получить участник за предложенную схему БД. Оценивается оптимальность принятых решений, соответствие исходной схеме данных, выполненные оптимизации.

Созданные участником представления оцениваются на БД малого и большого размеров. Представления участника не оцениваются, если после переноса данных выявлено отсутствие более 25% записей по какой-либо сущности.

Запросы к представлениям выполняются с заранее установленными фильтрами и сравниваются с эталонными ответами. Ошибки, невыполнение за конечное время и отличие от эталонного ответа оцениваются в 0 баллов.

Успешно выполненные запросы всех участников ранжируются по сложности плана запроса и времени выполнения в рамках каждого представления, где первое место получает максимум балов, а последнее - минимум.

До 25 баллов может получить участник за выполнение запросов к представлениям на БД малого размера.

До 25 баллов может получить участник за выполнение запросов к представлениям на БД большого размера. Участники, получившие 0 баллов за работу с каким-либо представлением на БД малого, автоматически получают 0 баллов за это же представление на БД большого размера.