ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

УХТИНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Кафедра ИСТ**

**Курсовой проект**

**Дисциплина: «Проект управления базами данных»**

**Тема:**

**«Разработка проекта управления базами данных для процесса «Учет ремонта и ТО автотранспорта»»**

**Выполнил**

**студент группы ИСТ-2-04**

**Петров М.В.**

**Проверила**

**доцент кафедры ИСТ, к. т. н.**

**Николаева Н.А.**

**Ухта 2008**

**Содержание**

|  |  |
| --- | --- |
| Введение  Краткое описание предметной области  Обоснование выбора SQL  Создание базы данных  Создание таблиц  Команды языка манипулирования данными  Запросы к базе данных  Хранимые процедуры  Триггеры  Заключение  Список литературы  Приложение |  |

**Введение**

Темой данного курсового проекта является реализация проекта базы данных для процесса «Учет ремонта и ТО автотранспорта».

Актуальность поставленной задачи состоит в необходимости разработать единую, наиболее полную структуру данных, необходимых для эффективной работы системы с целью организации учета ремонта и технического обслуживания автотранспорта.

Основной задачей курсового проекта, является создание таблиц базы данных и заполнение этих таблиц данными, обеспечение целостности базы данных путем создания декларативных ограничений и процедурных ограничений, разработка основных запросов и представлений для получения необходимой пользователю информации.

Целью курсового проекта, является реализация базы данных для автоматизированной системы, обслуживающей процесс учета ремонта и ТО автотранспорта.

Таким образом, для достижения цели данного проекта была проделана работа следующего содержания:

– краткое описание предметной области - в этом разделе курсового проекта описываются основные функции отдела реализации теплоснабжающей организации;

– обоснование выбора SQL - в этом разделе курсового проекта описан универсальный язык для создания, модификации и управления данными в реляционных базах данных;

– создание таблиц базы данных - в разделе описывается процесс создания таблиц, выбор типа данных, создание декларативных ограничений, описаны основные команды языка манипулирования данных;

– запросы к базе данных - в этом разделе курсового проекта описаны основные запросы к базе данных, которые позволяют получить необходимую для пользователя информацию;

– представления - в этом разделе курсового проекта описано создание шаблонов вывода информации, построенных на запросах к базе данных;

– триггеры - в этом разделе курсового проекта описан специальный тип хранимой процедуры, который позволяет обеспечить целостность данных, предотвращая несанкционированное или неправильное их изменение.

**Краткое описание предметной области**

Руководит всеми работами по ТО и ремонту подвижного состава начальник производства. Оперативное руководство производством работ на постах обслуживания и ремонта автомобилей осуществляет диспетчер производства, непосредственно подчиненный начальнику производства. Диспетчер производства заносит всю информацию по ремонту и ТО автотранспорта в журнал «Учет ремонта и технического обслуживания подвижного состава».

Диспетчеру в оперативном порядке подчиняются все работающие на постах ТО и ремонта автомобилей, а в отсутствие начальника производства на работе ему подчиняется весь коллектив производства.

При принятии решений по управлению производством диспетчер учитывает: общий объем работ, который необходимо выполнить, какие автомобили нуждаются в обслуживании и ремонте в первую очередь, содержание ремонта по каждому автомобилю, наличие производственного персонала и запасных частей и агрегатов, состояние технологического оборудования, время выхода автомобилей на линию и др.

Информацию о том, какие работы необходимо выполнять, диспетчер получает при знакомстве с незавершенным производством при приеме смены, по записям в листках учета, которые он получает по данным постов диагностики и сообщениям отдельных рабочих в процессе производства. По записям в листках учета он составляет оперативный план работы, в который записывает все автомобили, нуждающиеся в ТО и ремонте. Работы по ТО-1 проводятся в межсменное время, работы по ТО-2, в рабочее время с простоем для всех видов подвижного состава.

Если при выполнении обслуживания или ремонта рабочий обнаруживает необходимость выполнения такой работы, которая не записана в листке учета, он сообщает об этом своему руководителю или диспетчеру производства, выполняет работу, и ее записывают в листок.

Техническое обслуживание автомобилей в АТП производят на основе планов-графиков, составленных для каждого автомобиля исходя из среднесуточного пробега, нормативов периодичности ТО и условий эксплуатации на данном предприятии. Техническая служба вносит коррективы в план-график в зависимости от фактического пробега автомобиля и его состояния. Работы по ТО ведут бригадным или агрегатно-участковым методом.

Учет и отчетность по выполнению технического обслуживания и ремонта ведут по гаражным листам на каждый автомобиль, а также заявкам, составляемым водителем. При наличии на предприятии средств диагностирования выполняемые диагностические работы оформляют в виде диагностической карты определенной формы, где приводят проверяемые показатели автомобиля. Гаражные листы, заявки и диагностические карты утверждают мастера и начальники участков. Оформленные документы поступают в плановый отдел или центр управления производством, где их обрабатывают, систематизируют и используют в отчетности предприятия.

**Обоснование выбора SQL**

SQL - сокращение от Structured Query Language (Язык структурированных запросов) - это универсальный язык для создания, модификации и управления данными в реляционных базах данных.

Реляционная модель была предложена в 1970 году И.Ф. Коддом, работавшим в исследовательской лаборатории IBM в Сан-Хосе, Калифорния, и развивалась последующие десять лет в университетах и научных организациях. SQL - один из нескольких языков, выросших из этой идеи, в настоящее время практически полностью господствует в мире реляционных баз данных. Производители систем управления реляционными базами данных, первоначально использовавшие другие языки, сегодня полностью переориентировались на SQL.

SQL - это язык, на котором можно «разговаривать» с реляционными базами данных. Все системы управления базами данных предназначены для хранения и обработки информации. Реляционный подход к управлению базами данных основан на математической модели, использующей методы реляционной алгебры и реляционного исчисления.

Определение реляционной системы, так же, как и правила Кодда, требует чтобы весь диалог с базой данных велся на едином языке - иногда его называют общим подъязыком данных. В мире коммерческих систем управления базами данных такой язык получил название SQL.

SQL используется для манипуляции с данными (data manipulation) - выборки и модификации, определения данных (data definition) и администрирования данных (data administration). Любая операция по выборке, модификации, определению или администрированию выполняется с помощью оператора (statement) или команды (command) SQL.

Имеется две разновидности операции по манипуляции с данными - выборка данных (data retrieval) и модификация данных (data modification). Выборка - это поиск необходимых вам данных, а модификация означает добавление, удаление или изменение данных.

Операции по выборке (чаще называемые запросами (queries)) осуществляют поиск в базе данных, наиболее эффективно извлекают затребованную вами информацию и отображает ее. Во всех запросах SQL используется ключевое слово Select.

**Создание базы данных**

База данных – это именованная область на носителе данных, содержащая таблицы, индексы и другие объекты.

Для создания базы данных используется команда CREATE DATABASE

Создание базы данных "Ремонт и ТО автотранспорта"

USE master;

GO

IF DB\_ID (N'autorem') IS NOT NULL

DROP DATABASE autorem;

GO

CREATE DATABASE autorem ON PRIMARY

(SIZE = 20MB,Name='autorem',

Filename='D:\Programs\Autorem\Bases\autorem.mdf')

GO

**Создание таблиц**

Создав базу данных и подключившись к ней, можно начать создание таблиц. Таблицы являются основными строительными блоками базы данных. В них содержаться строки и столбцы данных. С помощью команд определения данных SQL можно создавать, удалять и манипулировать таблицами (добавлять, удалять, переставлять столбцы и менять их параметры).

В большинстве реализаций SQL таблицей владеет создавший ее пользователь, выдавая разрешения на ее использование другим пользователям. Чтобы создать таблицу, по меньшей мере, надо сделать следующее.

1. Задать имя таблицы.

2. Задать имена составляющих ее столбцов.

3. Определить тип данных для каждого столбца.

4. Определить ограничения для каждого столбца или таблицы.

Для создания таблицы используется команда Create Table.

Порядок создания таблиц: сначала создаются родительские (те, которые не содержат внешних ключей), затем дочерние таблиц. Если таблицы созданы и связаны правильно, то на диаграмме созданной в Enterprise Manager это будет выглядеть следующим образом (см. Приложение 1).

Назначение ограничений NULL/NOT NULL - разрешает или не разрешает неопределенность значений атрибутов; ограничение NULL (задается по умолчанию) явно указывает, что значения столбца могут быть не определены.

Ограничение UNIQUE - разрешает только уникальные значения атрибутов; Ограничение PRIMARY KEY - определяет первичный ключ отношения (в каждом отношении может иметь место только один первичный ключ, первичный ключ не может иметь NULL значений, одна и та же комбинация столбцов не может быть объявлена одновременно в качестве уникального и первичного ключа);

Ограничение FOREIGN KEY - определяет внешний ключ отношения (в одном отношении может быть несколько внешних ключей); организует ссылку по внешнему ключу на заданное в предложении REFERENCES отношение.

База данных "Ремонт и ТО автотранспорта" включает в себя следующие таблицы:

1. Автобаза (Autobase)
2. Справочник состояний (Condition)
3. Справочник технического обслуживания (Technical)
4. Тип марки (TypeMarka)
5. Марка (Marka)
6. Транспорт (Transport)
7. Состояние единицы (ConditionUnit)

*Таблица Автобаза (Autobase).*

Таблица содержит данные об автобазах. Каждая автобаза имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDAutobase)={NameAutobase, Address}, что аналогично R(Уникальный номер автобазы)={Название автобазы, Юридический адрес автобазы}. Атрибут IDAutobase должен быть уникальным и не равен нулю, т.к. является первичным ключом. Атрибут NameAutobase должен быть уникален, т.к. совпадающие названия автобаз состояний нарушают нормализацию таблицы базы данных, а также не равен NULL. Размер поля NameAutobase должен содержать символьные значения и максимальный среднестатистический размер атрибута равен 11 символов (к примеру, "Автобаза №1", "Автобаза №2"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 4 символов, имеем размер поля NameAutobase типа VARCHAR равным 15. Размер поля Address должен содержать символьные значения и максимальный среднестатистический размер атрибута равен 35 символов (например, "г. Воркута, ул. Автозаводская, 22"), следовательно, принимая во внимание необходимость резервной избыточности размера поля порядка 25 символов, имеем размер поля Address типа VARCHAR равным 60 (см. Таблица 1).

Таблица 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| Уникальный номер автобазы | IDAutobase | INT |  | NOT NULL, PRIMARY KEY |
| Название автобазы | NameAutobase | VARCHAR | 15 | NOT NULL, UNIQUE |
| Юридический адрес автобазы | Address | VARCHAR | 60 |  |

USE autorem;

GO

CREATE TABLE Autobase

(IDAutobase INT IDENTITY CONSTRAINT AutobasePrimary PRIMARY KEY,

NameAutobase VARCHAR (15) NOT NULL CONSTRAINT NameAutobaseUnique UNIQUE,

Address VARCHAR (60))

GO

*Справочник состояний (Condition)*

Таблица содержит данные о типе марки автотранспорта. Каждое состояние автотранспорта имеет уникальный номер, который является первичным ключом отношения.

Схема отношения имеет вид: R(IDCondition)={NameCondition}, что аналогично R(ID справочника состояний)={Название состояния}.

Атрибут IDCondition должен быть уникальным и не быть равным NULL, т.к. является первичным ключом. Атрибут NameCondition должен быть уникален, т.к. совпадающие названия состояний нарушают нормализацию таблицы базы данных, а также не равен NULL.

Размер поля NameCondition должен содержать символьные значения и максимальный среднестатистический размер атрибута не превышает 6 символов (например, "ремонт", "ТР", "ТО1", "ТО2", "СО"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 3 символов, имеем размер поля NameCondition типа VARCHAR равным 10 символам (см. Таблица 2).

Таблица 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| ID справочника состояний | IDCondition | INT |  | NOT NULL |
| Название состояния | NameCondition | VARCHAR | 10 | NOT NULL, UNIQUE |

USE autorem;

GO

CREATE TABLE Condition

(IDCondition INT IDENTITY CONSTRAINT ConditionPrimary PRIMARY KEY,

NameCondition VARCHAR (10) NOT NULL CONSTRAINT NameConditionUnique UNIQUE)

GO

*Справочник технического обслуживания (Technical)*

Таблица содержит данные о периодичности технического обслуживания подвижного состава. Каждый тип автотранспорта (легковые, автобусы, грузовые автобусы на базе грузовых автомобилей) имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDTechnical)={NameTechnical, TO1, TO2}, что аналогично R(ID справочника технического обслуживания)={Название типа автотранспорта, Километраж для TO1, Километраж для TO2}. Атрибут IDTechnical должен быть уникальным и не быть равным NULL, т.к. является первичным ключом. Атрибут NameTechnical должен быть уникален, т.к. совпадающие названия состояний нарушают нормализацию таблицы базы данных, а также не равен NULL. Размер поля NameTechnical содержит символьные значения и максимальный среднестатистический размер атрибута не превышает 46 символа (например, "грузовые автобусы на базе грузовых автомобилей", "легковые", "автобусы"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 13 символов, имеем размер поля NameTechnical типа VARCHAR равным 60 символам. Поле TO1 должно содержать числовые значения о пробеге автотранспорта до первого технического обслуживания, следовательно, поле ТО1 имеет тип INT. Поле TO2 должно содержать числовые значения о пробеге автотранспорта до второго технического обслуживания, следовательно, поле ТО2 имеет тип INT. (см. Таблица 3).

Таблица 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| ID справочника технического обслуживания | IDTechnical | INT |  | NOT NULL |
| Название типа автотранспорта | NameTechnical | VARCHAR | 60 | NOT NULL, UNIQUE |
| Километраж для TO1 | TO1 | INT |  |  |
| Километраж для TO2 | TO2 | INT |  |  |

CREATE TABLE Technical

(IDTechnical INT IDENTITY CONSTRAINT TechnicalPrimary PRIMARY KEY,

NameTechnical VARCHAR (60) NOT NULL CONSTRAINT NameTechnicalUnique UNIQUE,

TO1 INT,

TO2 INT)

*Тип марки (TypeMarka)*

Таблица содержит данные о типе марки автотранспорта. Каждая марка автотранспорта имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDTypeMarka)={IDTechnical, NameTypeMarka}, что аналогично R(Уникальный номер типа марки)={ID справочника технического обслуживания, Название типа марки}. Внешним ключом является IDTechnical (ID справочника технического обслуживания). Атрибут IDTypeMarka должен быть уникальным и не быть равным NULL, т.к. является первичным ключом отношения. Атрибут IDTechnical должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Атрибут NameTypeMarka должен быть уникален, т.к. совпадающие названия состояний нарушают нормализацию таблицы базы данных, а также не должен быть равен NULL. Размер поля NameTypeMarka должен содержать символьные значения и максимальный среднестатистический размер атрибута равен 20 символов (к примеру, "бортовая платформа", "экс.грейд.бульд.погр"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 15 символов, имеем размер поля NameTypeMarka типа VARCHAR равным 35 символам (см. Таблица 4).

Таблица 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| Уникальный номер типа марки | IDTypeMarka | INT |  | NOT NULL, PRIMARY KEY |
| ID справочника технического обслуживания | IDTechnical | INT |  | NOT NULL, FOREIGN KEY |
| Название типа марки | NameTypeMarka | VARCHAR | 35 | NOT NULL, UNIQUE |

USE autorem;

GO

CREATE TABLE TypeMarka

(IDTypeMarka INT IDENTITY CONSTRAINT TypeMarkaPrimary PRIMARY KEY,

IDTechnical INT NOT NULL,

NameTypeMarka VARCHAR (35) NOT NULL CONSTRAINT NameTypeMarkaUnique UNIQUE

CONSTRAINT TypeMarkaTechnicalForeign FOREIGN KEY (IDTechnical) REFERENCES Technical)

GO

*Таблица Марка (Marka).*

Таблица содержит данные о марках автотранспорта. Каждая марка имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDMarka)={IDTypeMarka, NameMarka, Capacity}, что аналогично R(Уникальный номер марки)={Уникальный номер типа марки, Название марки, Грузоподъемность}. Внешним ключом является IDTypeMarka (Уникальный номер типа марки). Атрибут IDMarka должен быть уникальным и не быть равным NULL, т.к. является первичным ключом. Атрибут IDTypeMarka должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Атрибут NameMarka должен быть не равен NULL. Размер поля NameMarka должен содержать символьные значения и максимальный среднестатистический размер атрибута равен 10 символов (к примеру, " КамАЗ-5320", "ЗиЛ-433100"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 5 символов, имеем размер поля NameCondition типа VARCHAR равным 15. Поле Capacity должен содержать дробные числовые значения о грузоподъемности автотранспорта, следовательно, поле Capacity имеет тип FLOAT (см. Таблица 5).

Таблица 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| Уникальный номер марки | IDMarka | INT |  | NOT NULL, PRIMARY KEY |
| Уникальный номер типа марки | IDTypeMarka | INT |  | NOT NULL, FOREIGN KEY |
| Название марки | NameMarka | VARCHAR | 15 | NOT NULL |
| Грузоподъемность | Capacity | FLOAT |  |  |

USE autorem;

GO

CREATE TABLE Marka

(IDMarka INT IDENTITY CONSTRAINT MarkaPrimary PRIMARY KEY,

IDTypeMarka INT NOT NULL,

NameMarka VARCHAR (15) NOT NULL,

Capacity FLOAT,

CONSTRAINT MarkaTypeMarkaForeign FOREIGN KEY (IDTypeMarka) REFERENCES TypeMarka)

GO

*Таблица Транспорт (Transport).*

Таблица содержит данные об автотранспорте. Каждый автотранспорт имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDTransport)={IDAutobase, IDMarka, GarageNumber, GosNumber}, что аналогично R(Уникальный номер единицы автотранспорта)={Уникальный номер автобазы, Уникальный номер марки, Гаражный номер, Госномер}. Внешними ключами являются IDAutobase (Уникальный номер автобазы), IDMarka(Уникальный номер марки). Атрибут IDTransport(Уникальный номер единицы автотранспорта) должен быть уникален и не равным NULL, т.к. является первичным ключом. Атрибут IDAutobase должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Атрибут IDTypeMarka должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Атрибут GarageNumber должен быть уникален, т.к. каждая единица автотранспорта имеет свой гаражный номер, а также не равен NULL. Атрибут GosNumber должен быть уникален, т.к. каждая единица автотранспорта имеет свой уникальный госномер, выданный в соответствующей ведомости. Размер поля GosNumber должен содержать символьные значения и максимальный среднестатистический размер атрибута равен 8 символам (к примеру, "В 990 АЕ", "КО 6718", "КУ 2664"), следовательно, принимая во внимание необходимость избыточности размера поля, имеем размер поля NameCondition типа VARCHAR равным 10. Размер поля GarageNumber должен содержать числовые значения о гаражном номере автотранспорта, следовательно, поле GarageNumber имеет тип INT (см. Таблица 6).

Таблица 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| Уникальный номер единицы автотранспорта | IDTransport | INT |  | NOT NULL, PRIMARY KEY |
| Уникальный номер автобазы | IDAutobase | INT |  | NOT NULL, FOREIGN KEY |
| Уникальный номер марки | IDMarka | INT |  | NOT NULL, FOREIGN KEY |
| Гаражный номер | GarageNumber | INT |  | NOT NULL, UNIQUE |
| Госномер | GosNumber | VARCHAR | 10 | UNIQUE |

USE autorem;

GO

CREATE TABLE Transport

(IDTransport INT IDENTITY CONSTRAINT TransportPrimary PRIMARY KEY,

IDAutobase INT,

IDMarka INT,

GarageNumber INT NOT NULL CONSTRAINT GarageNumberTransportUnique UNIQUE,

GosNumber VARCHAR(10) CONSTRAINT GosNumberTransportUnique UNIQUE,

CONSTRAINT TransportAutobaseForeign FOREIGN KEY (IDAutobase)REFERENCES Autobase,

CONSTRAINT TransportMarkaForeign FOREIGN KEY (IDMarka) REFERENCES Marka);

GO

*Состояние единицы (ConditionUnit)*

Таблица содержит данные обо всех состояниях автотранспорта. Каждый автотранспорт имеет уникальный номер, который является первичным ключом отношения. Схема отношения имеет вид: R(IDConditionUnit)={IDCondition, IDTransport, Working, Beginning, Theend}, что аналогично R(Уникальный номер работы)={Уникальный номер единицы автотранспорта, Уникальный номер состояния единицы, Описание работы, Начало состояния, Конец состояния}. Внешними ключами являются IDCondition (Уникальный номер состояния единицы), IDTransport (Уникальный номер единицы автотранспорта). Атрибут IDConditionUnit (Уникальный номер работы) должен быть уникален и не равным NULL, т.к. является первичным ключом. Атрибут IDCondition должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Атрибут IDTransport должен быть уникальным и не быть равным NULL, т.к. является внешним ключом отношения. Размер поля Working должен содержать символьные значения и максимальный среднестатистический размер атрибута должен равен 200 символов (например, "покраска переднего левого крыла; покраска крыши а/м; покраска передней левой двери; подготовка к покраске и покраска передней левой стойки; чистка хромированных деталей"), следовательно, принимая во внимание необходимость избыточности размера поля порядка 50 символов, имеем размер поля Working типа VARCHAR равным 250. Поля Beginning и Theend должны содержать даты постановки автотранспорта на ремонт и снятие с него, следовательно, типы этих полей определим как DateTime (см. Таблица 7).

Таблица 7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название атрибута | Имя поля | Тип поля | Размер | Ограничения |
| Уникальный номер работы | IDConditionUnit | INT |  | NOT NULL, PRIMARY KEY |
| Уникальный номер состояния единицы | IDCondition | INT |  | NOT NULL, FOREIGN KEY |
| Уникальный номер единицы автотранспорта | IDTransport | INT |  | NOT NULL, FOREIGN KEY |
| Описание работы | Working | VARCHAR | 250 |  |
| Начало состояния | Beginning | DateTime |  |  |
| Конец состояния | Theend | DateTime |  |  |

USE autorem;

GO

CREATE TABLE ConditionUnit

(IDConditionUnit INT IDENTITY CONSTRAINT ConditionUnitPrimary PRIMARY KEY,

IDCondition INT,

IDTransport INT,

Working VARCHAR(250),

Beginning DateTime,

Theend DateTime,

CONSTRAINT ConditionUnitConditionForeign FOREIGN KEY (IDCondition)REFERENCES Condition,

CONSTRAINT ConditionUnitTransportForeign FOREIGN KEY (IDTransport) REFERENCES Transport);

**Команды языка манипулирования данными**

Команды языка манипулирования данными (DML)

Три хорошо известные операции над кортежами:

1. Добавление, реализуется в SQL с помощью команды, INSERT

2. Правка, реализуется в SQL с помощью команды, UPDATE

3. Удаление, реализуется в SQL с помощью команды, DELETE Команда вставки INSERT используется для ввода новых строк в таблицу. Команда UPDATE - предназначена для редактирования данных в таблице. Команда DELETE - удаляет записи в таблице.

Команды языка определения данных (DDL)

Команда ALTER TABLE используется для внесения изменений в описание таблицы, в том числе для: добавления и изменения столбцов, добавления, разрешения, запрета и удаления ограничений.

Команда DROP TABLE используется для удаления таблицы.

Команда вставки − INSERT. Команда языка DML − INSERT используется для ввода новых строк в таблицу. Синтаксис команды:

INSERT INTO {<имя таблицы>[(<имя столбца> [псевдоним] [, …n]] |[<подзапрос>]}

VALUES (<значение>[,…n]);

При реализации команды INSERT необходимо отслеживать, чтобы последовательность данных в предложение VALUES, соответствовала порядку столбцов в таблице.

*Заполнение таблицы "Автобаза"*

INSERT INTO Autobase(NameAutobase,Address) VALUES('Автобаза №1','г.Воркута, ул.Транспортная,10');

INSERT INTO Autobase(NameAutobase,Address) VALUES('Автобаза №2','г.Воркута, ул.Комсомольская,11');

*Заполнение таблицы "Справочник состояний"*

INSERT INTO Condition (NameCondition) VALUES ('ТО1');

INSERT INTO Condition (NameCondition) VALUES ('ТО2');

INSERT INTO Condition (NameCondition) VALUES ('СО');

INSERT INTO Condition (NameCondition) VALUES ('ТР');

*Заполнение таблицы "Справочник технического обслуживания"*

INSERT INTO Technical (NameTechnical,TO1,TO2) VALUES ('легковые', 3500, 14000);

INSERT INTO Technical (NameTechnical, TO1, TO2) VALUES ('автобусы',2600,13000);

INSERT INTO Technical (NameTechnical,TO1,TO2) VALUES ('грузовые, автобусы на базе грузовых автомобилей',2200,11000);

*Заполнение таблицы "Тип марок автотранспорта"*

INSERT INTO TypeMarka (NameTypeMarka, IDTechnical) VALUES ('легковая',1);

INSERT INTO TypeMarka (NameTypeMarka, IDTechnical) VALUES ('мазутовоз',3);

INSERT INTO TypeMarka (NameTypeMarka, IDTechnical) VALUES ('молокоцистерна',3);

INSERT INTO TypeMarka (NameTypeMarka,IDTechnical) VALUES ('поливомоечная',3);

INSERT INTO TypeMarka (NameTypeMarka, IDTechnical) VALUES ('самосвал',3);

INSERT INTO TypeMarka (NameTypeMarka, IDTechnical) VALUES ('спец.фургон',3);

*Заполнение таблицы "Марка автотранспорта"*

INSERT INTO Marka (IDTypeMarka, NameMarka, Capacity) VALUES (1,'ГАЗ-31029',0.00);

INSERT INTO Marka (IDTypeMarka, NameMarka, Capacity) VALUES(1, УАЗ-2206',0.00);

INSERT INTO Marka (IDTypeMarka, NameMarka, Capacity) VALUES (22,'КАВЗуд ЛАЗ',0.00);

*Заполнение таблицы "Автотранспорт"*

INSERT INTO Transport (IDAutobase, IDMarka, Garage Number, Gos Number) VALUES (1,294, 502,'11ко3813');

INSERT INTO Transport (IDAutobase, IDMarka, GarageNumber, GosNumber) VALUES(1,294, 837,'11ко3866');

INSERT INTO Transport (IDAutobase, IDMarka, Garage Number, GosNumber) VALUES(1,483, 125,'14-51КМО');

INSERT INTO Transport (IDAutobase, IDMarka, GarageNumber, GosNumber) VALUES (1,483, 152,'14-52КМО');

*Заполнение таблицы "Состояние единицы"*

INSERT INTO Condition Unit (IDCondition, IDTransport, Working, Beginning, Theend) VALUES (2,1,'покраска переднего левого крыла; покраска крыши а/м','2008.09.15','2008.09.16');

INSERT INTO ConditionUnit (IDCondition, IDTransport, Working, Beginning, Theend) VALUES (2,2,'покраска передней левой двери; подготовка к покраске и покраска передней левой стойки','2008.09.15','2008.09.16');

INSERT INTO ConditionUnit (IDCondition, IDTransport, Working, Beginning, Theend) VALUES (2,3,'чистка хромированных деталей','2008.09.15','2008.09.16');

**Запросы к базе данных**

Команда SELECT позволяет реализовывать все операторы реляционной алгебры над отношениями (таблицами) базы данных. Обязательными предложениями команды SELECT являются предложения SELECT и FROM. В самой простой форме, команда SELECT используется для того, чтобы извлечь информацию из таблицы. Не обязательно использовать все предложения команды, но обязательно соблюдать их порядок следования, то есть предложение ORDER BY всегда завершает команду SELECT, а предложение HAVING всегда стоит после GROUP BY, которое, в свою очередь, не может предшествовать предложению WHERE и т.п.

Синтаксис команды SELECT:

SELECT [DISTINCT] \*|<столбец> [<псевдоним>] [,<групповая функция>] [,…n]

FROM <таблица>[, …n]| (<подзапрос>)

[WHERE <условие>]

[GROUP BY<выражение группировки]

[HAVING <условие отбора группы>]

[ORDER BY < столбец >[,…n]]

Команда SELECT позволяет извлечь определенную информацию из таблицы. Например, просмотреть содержимое одного или нескольких столбцов, присвоив столбцам вывода имена, отличные от имен атрибутов в таблице, но более информативные. Чтобы вывести содержание всех столбцов, можно заменить их перечисление знаком \*. В этом случае столбцы будут выведены в том порядке, в котором они следуют в таблице. Если вы хотите изменить порядок, то придется перечислить имена столбцов в нужном порядке.

Предложение FROM используется для указания перечня таблиц, используемых в запросе и условия их соединения.

**Выбор всех автобаз**

Select \* from AutoBase

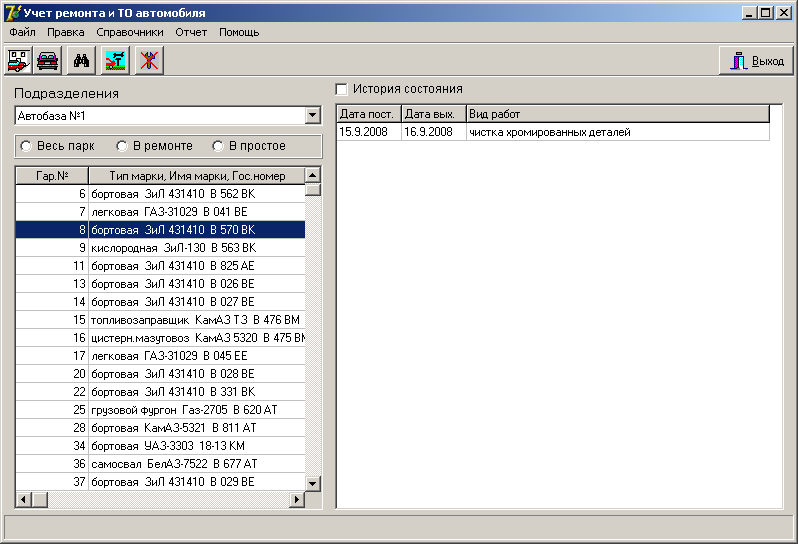
Выбор Гаражного номера, Госномера из таблицы "Автотранспорт", Имя марки из таблицы "Марка автотранспорта", Тип марки из таблицы "Тип марки автотранспорта"

Select T.IDTransport, T.GarageNumber[Гар.№], TM.NameTypeMarka+' '+M.NameMarka+' '+T.GosNumber[Тип марки, Имя марки, Гос.номер] from Transport T INNER JOIN Marka M ON T.IDmarka=M.IDmarka INNER JOIN TypeMarka TM ON M.IDTypeMarka=TM.IDTypeMarka where (T.IDAutobase=:IDAutoBase) ORDER BY T.GarageNumber;

Выбор Даты постановки в ремонт, Дата снятия с ремонта, Вид работ

Select LTrim(Str(Day(Beginning)))+'.'+LTrim(Str(Month(Beginning)))+'.' +LTrim(Str(Year(Beginning)))[Дата пост.],LTrim(Str(Day(Theend)))+'.' +LTrim(Str(Month(Theend)))+'.'+LTrim(Str(Year(Theend)))[Дата вых.], Working[Вид работ] from ConditionUnit where (IDTransport=:IDTransport);

Согласно приведенным выше запросам имеем некоторые выходные данные:



**Хранимые процедуры**

Хранимая процедура ещё один объект базы данных, которая представляет собой набор откомпилированных операторов SQL. Хранимая процедура не содержит информации из базы данных, но содержит ссылки на базовые таблицы, где хранятся нужные данные. Хранимые процедуры позволяют выделять какие-либо правила в отдельную структуру, которые затем могут использоваться многими приложениями.

В хранимые процедуры вводятся аргументы, возвращаются результирующие наборы данных. Если хранимая процедура не является триггером, то она вызывается приложением явно.

При создании процедур необходимо придерживаться следующих правил: во время выполнения хранимой процедуры все объекты, на которые она ссылается, должны присутствовать в базе данных. В хранимых процедурах нельзя применять операторы создания объектов: CREATE PROCEDURE, CREATE TRIGGER, CREATE VIEW.

Синтаксис команды:

CREATE PROC[EDURE ] имя\_процедуры [; число ]

[ { @параметр тип\_данных }

[ VARYING ] [ = значение\_по\_умолчанию ] [ OUTPUT ] ] [,...n ]

[ WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION } ]

AS оператор\_SQL [ ...n ]

OUTPUT – означает, что соответствующий параметр может быть использован для возвращения данных из хранимой процедуры.

VARYING – определяет, что выходным параметром будет результирующее множество (используется совместно с параметром OUTPUT).

RECOMPILE – предписывает выполнять перекомпиляцию кода процедуры при ее запуске.

ENCRUPTION – предписывает серверу выполнить кодирование процедуры, сделав ее содержимое недоступным для пользователя. Автору процедуры при этом надо позаботиться о наличии у него исходного текста процедуры.

AS – индикатор начала собственно кода процедуры.

Вызов процедуры (исключение составляют триггеры) осуществляется по команде EXEC.

Синтаксис команды EXEC

EXEC <Имя процедуры> [<список параметров>]

- Создадим хранимую процедуру для таблицы "Марка автотранспорта"

CREATE PROC PMarkaINSERTFirst

@IDTypeMarka INT,

@NameMarka VARCHAR (15),

@Capacity FLOAT

AS

INSERT INTO Marka(IDTypeMarka,NameMarka,Capacity)VALUES( @IDTypeMarka, @NameMarka,@Capacity);

Теперь заполнение таблицы "Марка автотранспорта" через хранимые процедуры будет иметь следующий вид:

EXEC PMarkaINSERTFirst 1,'ГАЗ-31029',0.00

EXEC PMarkaINSERTFirst 22,'КАВЗ,ПАЗ',0.00

EXEC PMarkaINSERTFirst 22,'КАВЗуд ЛАЗ',0.00

EXEC PMarkaINSERTFirst 22,'КАВЗуд ЛАЗ',0.00

EXEC PMarkaINSERTFirst 22,'КАВЗ,ПАЗ',0.00

Создание хранимые процедуры для запросов Select

Создание хранимой процедуры для таблицы "Марка автотранспорта"

- Выбор всех автобаз

CREATE PROC PAutoBaseSelect

AS

Select \* from AutoBase

Теперь выборка всех автобаз будет иметь следующий вид

Exec PAutoBaseSelect;

Выбор Гаражного номера, Госномера из таблицы "Автотранспорт", Имя марки из таблицы "Марка автотранспорта", Тип марки из таблицы "Тип марки автотранспорта"

CREATE PROC PTransportSelect1

@IDAutoBase INT

AS

Select T.GarageNumber[Гар.№], T.GosNumber[Гос.номер], M.NameMarka[Имя марки], TM.NameTypeMarka[Тип марки] from Transport T INNER JOIN Marka M ON T.IDmarka=M.IDmarka

INNER JOIN TypeMarka TM ON M.IDTypeMarka=TM.IDTypeMarka where (T.IDAutobase=@IDAutoBase);

Теперь выбор Гаражного номера, Госномера из таблицы "Автотранспорт", Имя марки из таблицы "Марка автотранспорта", Тип марки из таблицы "Тип марки автотранспорта" будет иметь следующий вид

Exec PTransportSelect1 1;

Выбор Даты постановки в ремонт, Дата снятия с ремонта, Вид работ

CREATE PROC PConditionUnit1

@IDTransport INT

AS

Select LTrim(Str(Day(Beginning)))+'.'+LTrim(Str(Month(Beginning)))+'.'+LTrim(Str(Year(Beginning)))[Дата пост.],

LTrim(Str(Day(Theend)))+'.'+LTrim(Str(Month(Theend)))+'.'+LTrim(Str(Year(Theend)))[Дата вых.], Working[Вид работ]

from ConditionUnit where (IDTransport=@IDTransport);

Теперь Выбор Даты постановки в ремонт, Дата снятия с ремонта, Вид работ примет следующий вид

Exec PConditionUnit1 1;

**Триггеры**

Триггер – это специальный тип хранимой процедуры, которая автоматически выполняется при возникновении некоторого события (попытке выполнить операции удаления, добавления, редактирования). Как и любой объект базы данных, триггер создается с помощью команды CREATE

### Cинтаксис команды создания триггера

CREATE TRIGGER <имя триггера>

ON <имя представления или таблицы>

[WITH ENCRYPTION]

{{{FOR|AFTER}<[DELETE][,][INSERT] [,][UPDATE]>}|INSTEAD OF}

[WITH APPEND]

[NOT FOR REPLICATION]

AS

<SQL –ОПЕРАТОР

….

….

….>,

где

ON − имя объекта, для которого триггер используется.

WITH ENCRYPTION – кодирует текст представления.

WITH APPEND – используется для совместимости с версией 6.5

NOT FOR REPLICATION – меняет правила запуска триггера. Такой триггер не будет стартовать при выполнении над таблицей операций, связанных с репликацией данных.

При срабатывании триггера создаются таблицы INSERTED, DELETED. INSERTED – для хранения добавленных записей, DELETED – для хранения удаленных записей. Таблицы видимы только для триггера и существуют только при выполнении триггера.

Создание триггера для таблицы "Состояние единицы".Проверка на ввод пустой даты окончания ТО или ремонта и замена ее на 9999.01.01

CREATE TRIGGER ConditionUnitTerm

ON ConditionUnit

FOR INSERT,UPDATE

AS

IF EXISTS

( SELECT 'TRUE' FROM ConditionUnit WHERE Theend='1900.01.01')

BEGIN

UPDATE ConditionUnit SET Theend='9999.01.01' WHERE Theend='1900.01.01'

END

**Заключение**

В результате курсового проекта была реализована база данных для процесса «Учет ремонта и ТО автотранспорта». Были созданы таблицы, в которых были определены ограничения на уровне столбцов и на уровне самой таблицы, и занесены необходимые данные. Изучение предметной области позволило выделить и реализовать основные запросы и представления к базе данных.

Для реализации целостности базы данных был созданы триггер, изменяющий введенную дату окончания ремонта или ТО.

Реализованная база данных позволит в дальнейшем хранить необходимую информацию, добавлять данные и редактировать их. Эта база данных может быть использована в дальнейшем для разработки информационной системы Учет ремонта и ТО автотранспорта», с целью эффективной организации работы отдела ПТО УАТХ.

**Список литературы**

1. Томас Конноли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика: Пер. с англ. М.:Вильямс> 2001.-1440с.
2. К. Дж. Дейт. Введение в системы баз данных.: Пер. с англ. М.:Вильямс? 2001.-1072с.
3. Г.Н. Калянов. Консалтинг в автоматизации бизнес процессов. М.: Горячая линия - Телеком, 2002. - с.320.:ил.

**Приложение**

Приложение

