**Эволюция развития баз данных**

В истории развития вычислительной техники можно проследить 2 направления ее использования: численные расчеты, информационные системы.

Основные операции в базах данных – CRUD(create update delete)

Информационная система – совокупность средств (технических, программных, информационных, организационных, методов и персонала), предназначенных для ввода, хранения, обработки и выдачи инфо предоставленных в интересах поставленной задачи.

В первых ИС инфо хранилась в виде послед. файлов на магнитных лентах (устройствах с последовательным доступом).

Требования ИС вызвали появление устройств на магнитных дисках.

ИС построенные на посл. файлах имели ряд недостатков: избыточность данных, слабый контроль, зависимость программ от данных (при изменении структуры файла необходимо внести изменения во все программы, работающие с этим файлом), огромные затраты труда программистов

Эти недостатки послужили толчком, который заставил разработчиков ИС искать новый подход к управлению инфо. Этот подход был реализован в рамках новых программных средств, названных СУБД, а сами хранилища БД

БД (Data Base) – совокупность данных, отражающая состояние объектов и их отношение в рассматриваемой предметной области.

СУБД (Data Base Management System) – совокупность программных и языковых средств, предназначенных для создания ведения и совместного использования многими пользователями.

Предметная область – область знаний или человеческой деятельности для которой проектируется БД.

Этапы развития баз данных

1. Базы данных на больших ЭВМ
2. Эпоха персональных компьютеров
3. Распределенные БД
4. Новые направления(NOSQL)
5. Иерархические СУБД (1968 IBM)  
   Узлы хранят данные, а указатели указывают о расположении данных  
   В иерархической наблюдается связь 1(предок):Многим(потомкам)  
   Разузлование (планирование производства)  
   Достоинства: высокое быстродействие, простая структура.  
   Недостатки: зависимость программ от данных, иерархическая модель применима к данным которые связаны между собой иерархически.
6. Сетевые СУБД (1971 CODASYL)  
   У каждого предка может быть 0,1 или более потомков, так же и у потомков.  
   Появление сетевых СУБД было обусловлено появлением сетевых заказов.

Достоинства: Относительно высокое быстродействие, гибкость.

Недостатки: сложная структура, зависимость программ от данных, требуется mainframe (мощная вычислительная машина).

Модель – совокупность структур данных и операций их обработки.

**Реляционная модель** (от англ. Relation - отношение).

Основоположником является англ. математик Кодд (Codd).

Он предложил использовать двумерную таблицу для хранения данных и использовать математический аппарат для обработки реляционных таблиц.

Не любая 2-мерная таблица не является реляционной.

Свойства реляционной таблицы:

1. В каждой ячейке хранится неделимое значение.
2. Все столбцы однородные.
3. Дубликаты строк отсутствуют
4. Каждый столбец имеет уникальное имя
5. Возможна произвольная перестановка как строк, так и столбцов

|  |  |
| --- | --- |
| **Реляционный термин** | **Определение** |
| Схема базы данных | Совокупность таблиц и связей между ними. |
| Схема отношения | Совокупность заголовков столбцов |
| Тело отношения | Совокупность записей или строк |
| Атрибут | Название столбца или поля |
| Кортеж | Строка или запись |
| Степень отношения | Кол-во столбцов |
| Мощность | Кол-во записей (строк) |
| Домен | Совокупность допустимых значений атрибута |
| Первичный ключ (Primary Key) | Минимальный набор атрибутов по которым можно однозначно выбрать каждую запись |
| Внешний ключ (Foreign Key) | Поле в зависимой таблице, которое соединяется с первичным ключом |

Первичный ключ обладает свойствами: уникальность, минимальность (ключ минимален если при удалении хотя бы 1 атрибута из состава первичного ключа, он теряет свойство уникальности).

Простой первичный ключ – состоит из 1 атрибута, а составной – из нескольких.

Достоинства реляционной модели: простота, используется строгий математический аппарат для обработки таблицы, стандарт языка SQL, поддержка целостности данных.

Недостатки: невысокая производительность, не всякую предметную область можно представить в виде реляционной модели, трудность понимания бд из-за появления большого кол-ва таблиц.

Целостность данных – непротиворечивость данных в бд.

**Подготовиться к проверочной работе по пройденным темам!!!**

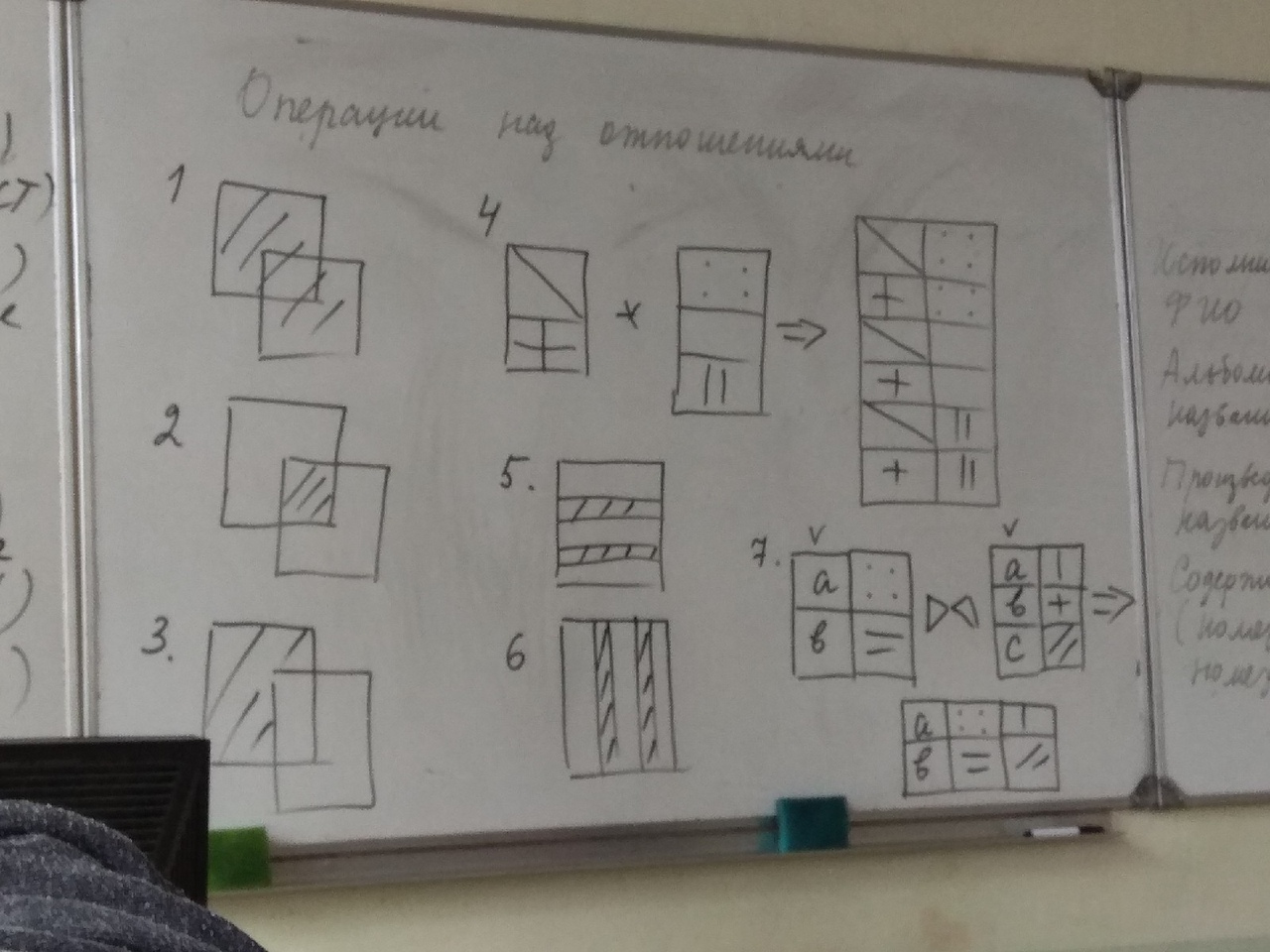
Операции над отношениями

Кодд не только разработал, но также предложил языки: реляционная алгебра, реляционное исчисление для обработки таблиц. Оба языка основаны на математике. Реляционная алгебра – процедурный язык. В ней имеется пошаговый подход к созданию запроса. Реляционное исчисление – непроцедурный язык. Основан на исчислении предикатов.

Предикат – истинностная функция с аргументами, в которой при подстановке значений аргументов функция возвращает значение true или false.

В реляционном исчислении запрос создается путем определения таблицы за 1 шаг. Оба языка взаимно эквивалентны.

1. Объединение (UNION)
2. Пересечение (INTERSECT)
3. Разность (EXCEPT)
4. Декартово произведение (CROSS JOIN)
5. Выборка (SELECT)
6. Проекция (PROJECT)
7. Соединение (JOIN)
8. Деление (DIVISION)



Для выполнения операции соединения JOIN необходимо чтобы у таблиц был хотя бы 1 общий атрибут. Типы данных, соединяемых атрибутами, должны совпадать. Соединение записей осуществляется по совпадающим значениям соединяемых атрибутов.

R1

|  |  |  |
| --- | --- | --- |
| **A** | **b** | **C** |
| 1 | 2 | 3 |
| 4 | 5 | 6 |

R2

|  |  |  |
| --- | --- | --- |
| **A** | **b** | **C** |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

R3

|  |  |
| --- | --- |
| **A** | **E** |
| 1 | K |
| 1 | E |
| 7 | M |

|  |  |  |
| --- | --- | --- |
| **A** | **b** | **c** |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

1. =

|  |  |  |
| --- | --- | --- |
| **A** | **b** | **C** |
| 4 | 5 | 6 |

1. \

|  |  |  |
| --- | --- | --- |
| **A** | **b** | **c** |
| 1 | 2 | 3 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **B** | **c** | **d** | **e** | **f** |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 2 | 3 | 7 | 8 | 9 |
| 4 | 5 | 6 | 4 | 5 | 6 |
| 4 | 5 | 6 | 7 | 8 | 9 |



|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **B** | **c** | **d** |
| 1 | 2 | 3 | K |
| 1 | 2 | 3 | E |

Задание : в базе данных хранятся данные R1-список абитуриентов сдававших экзамены на подготовительных курсах, R2- список абитуриентов сдававших на   
R3 – список абитуриентов принятых в колледж. Используя операции над отношениями определить:

1. Список абитуриентов которые поступали 2 раза и не поступили (есть и в р1 и в р2 но нет в р3).
2. Поступили с первого раза(есть в р1 и в р3 но нет в р2).
3. Поступили со второго раза (есть в р2 и в р3 но нет в р1)
4. Поступали 1 раз и не поступили (есть или в р1 или в р2 но нет тогда в других)

Решение:

1. \R2)

**Типы связи в информационной модели**

1 запись соответствует 1 экземпляру информационного объекта.

3 типа связи: 1:1, 1:М, М:М.

Между 2-мя объектами существует связь 1:1, если 1 экземпляр A имеет связь с экземпляром B, но 1 экземпляр B связан в точности с 1 экземпляром A.

Между 2-мя объектами существует связь 1:М, если 1-му экземпляру объекта А соответствует 0,1 или несколько экземпляров объектов B, но 1 экземпляр объекта В связан в точности с 1 экземпляром объекта A.

Между 2-мя наблюдается связь М:М, если 1-му экземпляру объекта А соответствует 0,1 или несколько экземпляров объектов B, и 1 экземпляр объекта В связан в точности с 1 или более экземплярами объекта A.

В реляционной БД связь М:М реализуется через промежуточную таблицу.

(В практической потребуется программа ERassistant)

Определение типа связи между объектами.

1. Выбрать где главная, а где зависимая таблица. Главная таблица это та, которой не требуется другая для полной идентификации таблицы.
2. Берем 1 запись в главной таблице. Смотрим сколько ей потенциально может соответствовать записей в зависимой таблице. Если не более 1, то связь 1:1.

Задание: определить модель БД,

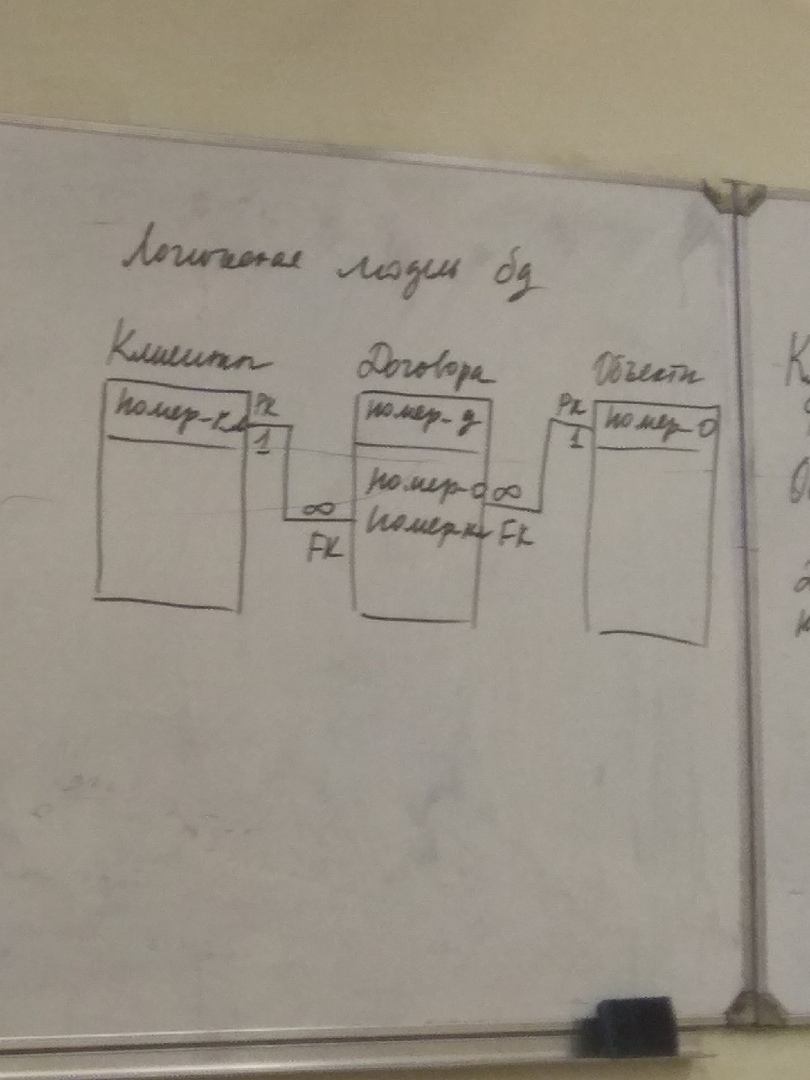
Клиенты(номер\_кл, Ф, И, О,…)

Объекты(номер объекта, адрес, цена в сутки)

Договора(номер\_д, дата, номер\_кл, номер\_об, к-во дней).

Решение:

Логическая модель БД:



1 клиент может заключить 0, 1 или более договоров. Но конкретный договор заключает 1 клиент.

Задание 2:Певцы(номер\_певца, ф, и, о);альбомы(номер\_альбома, дата, название); песни(номер\_песни, название); содержание альбома(номер\_певца, номер\_альбома, номер\_песни)

Решение:

певцы

альбомы

песни

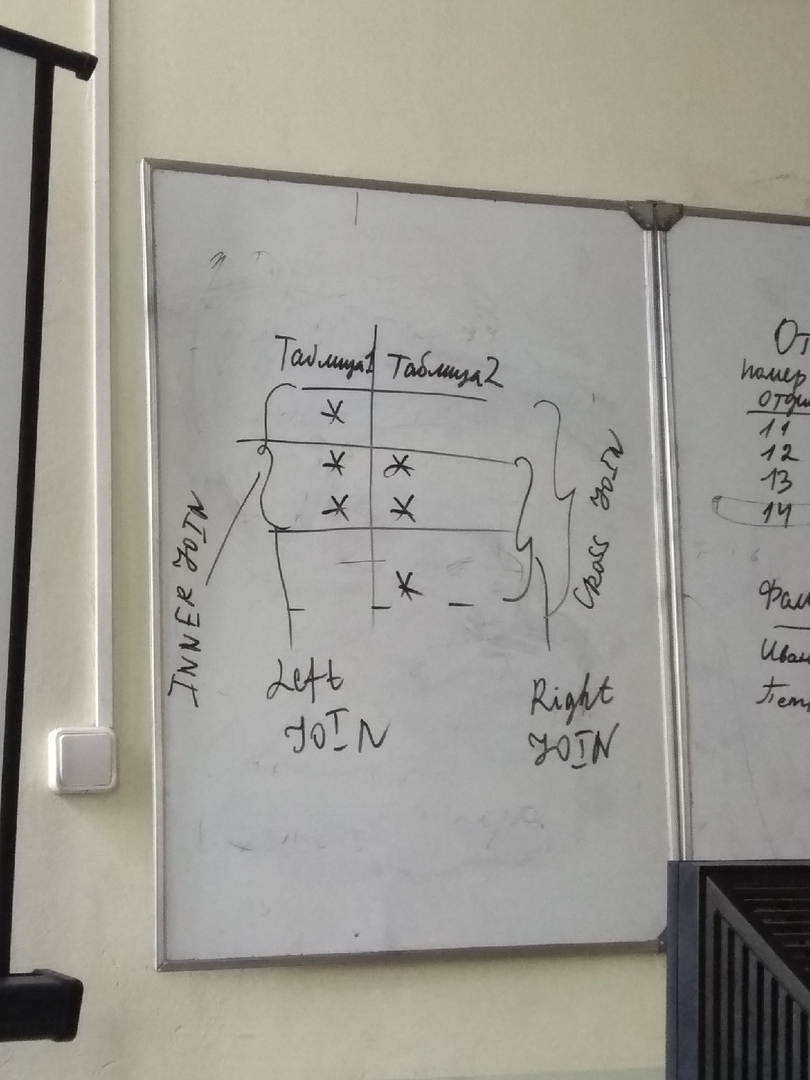
содержание

**Целостность данных.**

Целостность данных – непротиворечивость данных в БД.

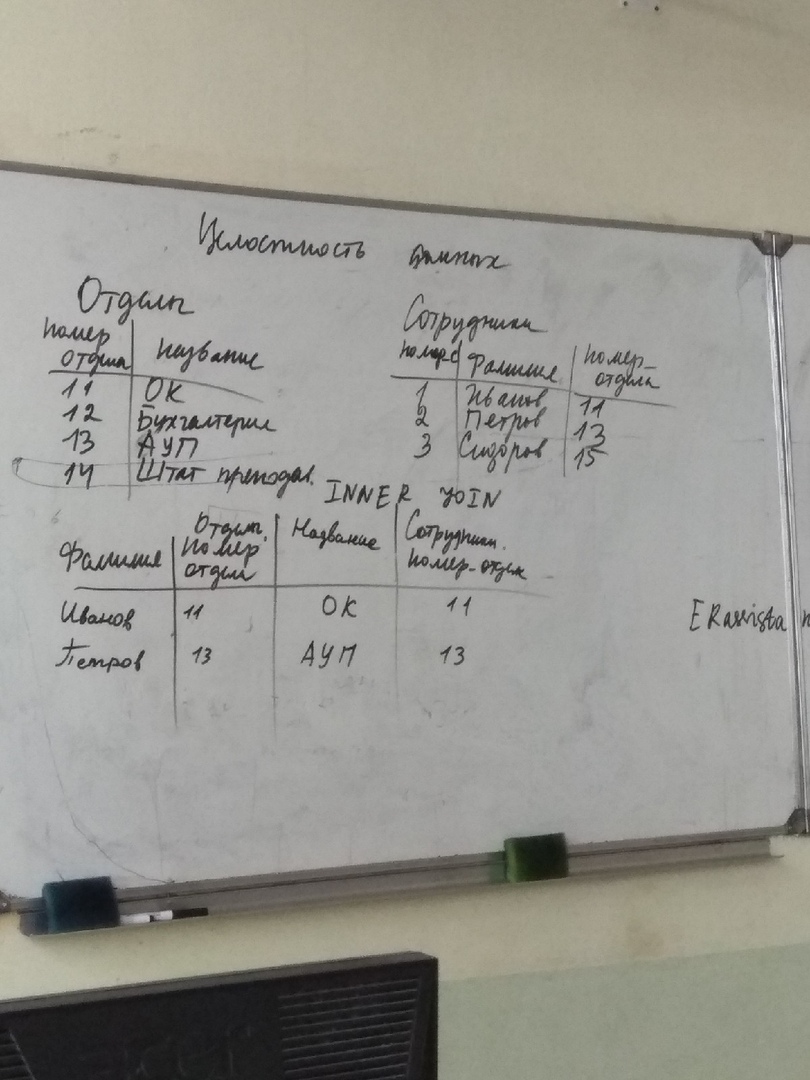
Типы соединений между таблицами:

1. Симметричное (INNER JOIN)
2. Несимметричное
3. Левое (LEFT JOIN)
4. Правое (RIGHT JOIN)
5. Декартово произведение (CROSS JOIN)



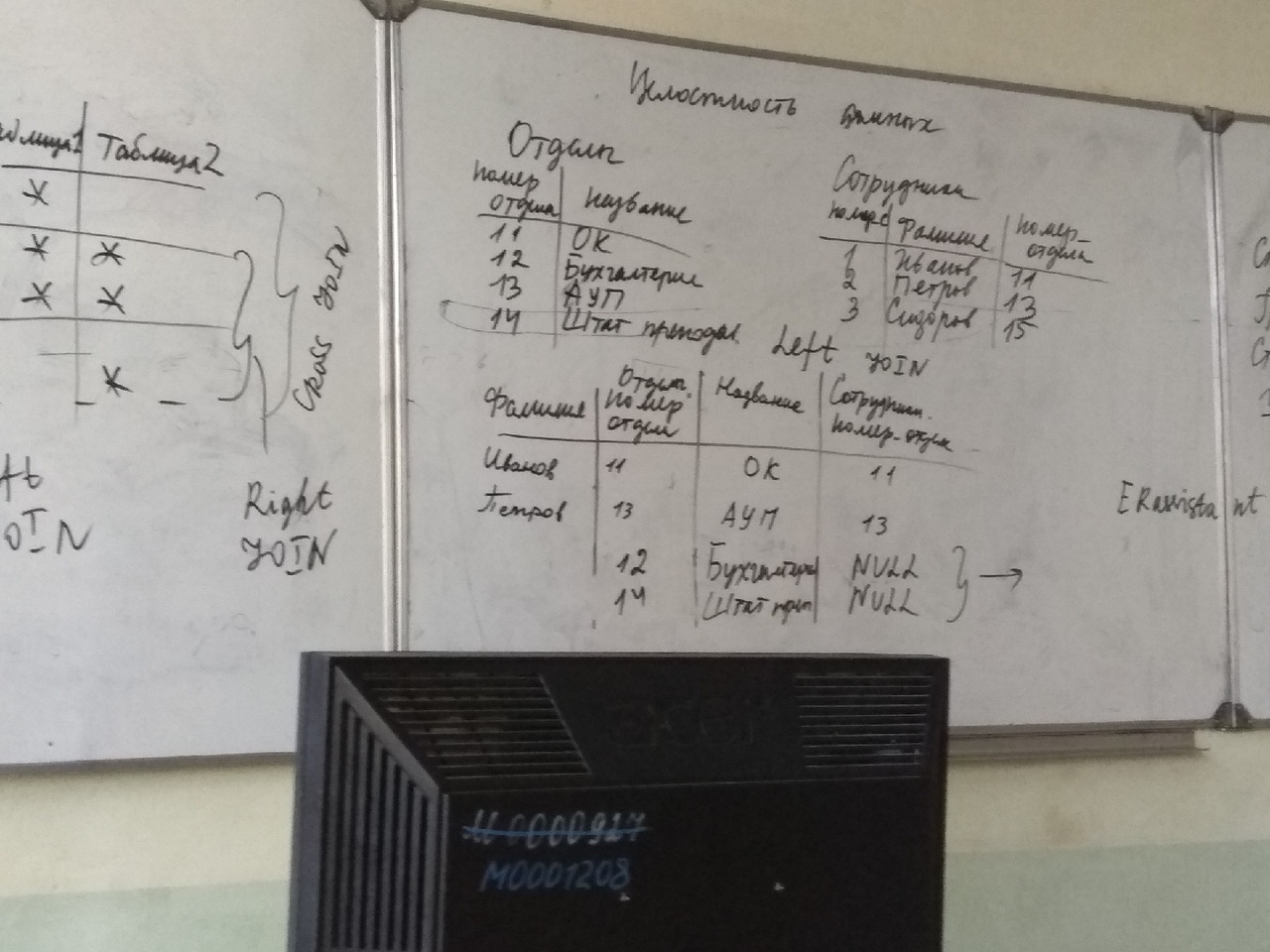
№1) Создать запрос в котором вывести фам. Номер отдела и название отдела, в котором работает работник.

INNER JOIN



№2) Вывести название отделов в котором не работает ни один сотрудник.

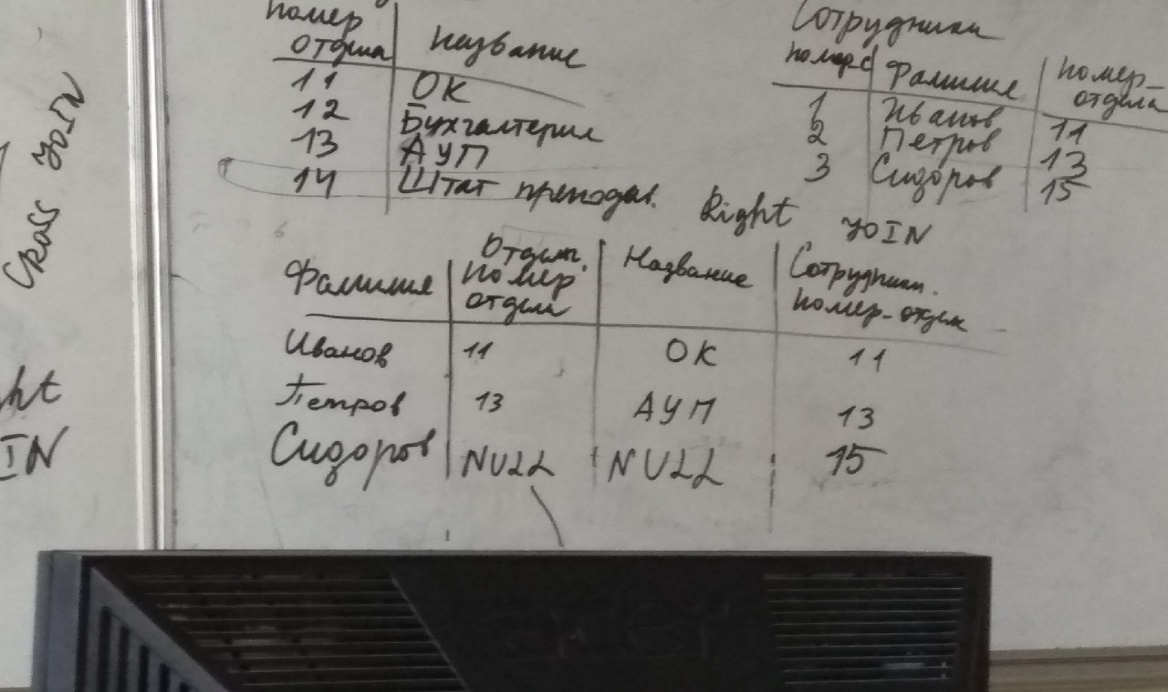
LEFT JOIN



Все записи из левой таблицы и из правой только те в которых значения атрибутов совпадают.

№3) Создать запрос в котором вывести фамилии работников, которые работают в несуществующих отделах.

RIGHT JOIN



В данных таблицах нарушена целостность данных, поэтому надо изменить номер отдела у сотрудника на существующий, либо изменить номер в самих отделах.

Различают категорную (никакой первичных ключ не может принимать пустое значение), доменную (при выборе типа данных происходит ограничение значения атрибута), ссылочную (значение внешнего ключа в зависимой таблице должно соответствовать значению первичного ключа в главной таблице) целостность данных.

Ссылочная целостность устанавливается при создании диаграммы БД или в команде CREATE TABLE при создании внешних ключей.

Частным случаем ссылочной целостности является каскадное обновление и каскадное удаление связанных записей.

Каскадное обновление – при обновлении первичного ключа в главной таблице происходит обновление внешнего ключа во всех связанных записях.

Каскадное удаление – при удалении записи в главной таблице происходит удаление записей во всех внешних таблицах.

**Организация запросов в БД на SQL**

SQL – **S**tructural **Q**uery **L**anguage (ранее SEQUEL)

Язык появился в 1970, разработан фирмой IBM.

Виды команд:

I группа команд

1. DDL   
   CREATE, DROP, ALTER
2. DML  
   SELECT,UPDATE, DELETE, INSERT
3. DCL  
   GRANT  
   REVOCE  
   DENY
4. TCL  
   Begin tran  
   COMMIT  
   ROLLBACK

1-4 команды системных аналитиков

II группа команд для программистов.

Когда мы говорим об SQL мы предполагаем, что используется реляционная модель, используется клиент-серверная архитектура и многопользовательская среда.

Сервер БД: MS SQL Server, MYSQL, DB2, ORACLE, PostgreSQL.

DDL

Create – создать

Drop – удалить

Alter – изменить

Синтаксис

Команды пишутся прописными(заглавными) а записи и названия строчными.

Файлы БД: \*.mdf (данные); \*.ldf (журнал транзакций).

Каждый пакет команд разделяется с помощью GO

Команда создания таблиц 🡪 CREATE TABLE

[] означают необязательные параметры.

USE Master

IF EXISTS (SELECT \* FROM Sysdatabases (sysobjects – для объектов бд) WHERE

NAME = ‘test’

DROP DATABASE test) – для удаления и повторного создания бд

CREATE DATABASE test

ON PRIMARY

(NAME = testdata,

FILENAME = ‘C:\Users\...\testdata.mdt’,

SIZE = 3 MB, MAXSIZE = 100 MB,

FILEGROWTH = 10 MB)

LOG ON

(NAME = testlog,

FILENAME = ‘C:\Users\...\testlog.ldf’,

SIZE = 3 MB, MAXSIZE = 100 MB,

FILEGROWTH = 10 MB)

CREATE TABLE имя\_таблицы

(имя\_столбца1, тип данных, [ограничения], имя\_столбца2…)

Ограничения – механизм, обеспечивающий автоматический контроль данных установленным условием.

1. NULL/NOT NULL
2. DEFAULT
3. PRIMARY KEY
4. UNIQUE
5. CHECK
6. FOREIGN KEY

Пример

CREATE TABLE departments

(department\_id INT PRIMARY KEY

CHECK (department\_id>!0),

department\_name NVARCHAR(30) NOT NULL DEFAULT ‘OK’

)

Создать таблицу сотрудники с полями номер сотрудника>0, дата, фИ(not null), адрес почты(unique), номер телефона(default +7916 – 000 – 00 – 00), оклад(>120k), номер отдела(foreign key от departments)

Ответ

CREATE TABLE employers (employer\_id INT PRIMARY KEY CHECK(employer\_id>0),

birth\_date DATE,

emp\_name NVARCHAR(30) NOT NULL,

last\_name NVARCHAR(30) NOT NULL,

e\_mail NVARCHAR(30) UNIQUE,

telephone NCHAR(12) CHECK(telephone LIKE ‘+7[0-9][0-9][0-9]…’ DEFAULT ‘+79160000000’

salary MONEY DEFAULT 50000 CHECK (salary>=20000 AND salary <= 100000))

department\_id INT FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

Команды SQL

**SELECT**

Синтаксис команды SELECT

SELECT [предикат] список полей

FROM таблица1[,…]

[WHERE условия отбора.]

[GROUP BY …]

[HAVING условия отбора..]

[ORDER BY …]

Список полей:

\*| таблица. \*|

[таблица.]поле\_1

[AS псевдоним\_1][, таблица.]поле\_2[AS псевдоним\_2][, …]]

Выбор всех столбцов:

SELECT \* FROM таблица

Выбор конкретных столбцов:

SELECT фамилия, имя,[дата\_рождения] FROM таблица

Использование псевдонимов:

SELECT fam as[фамилия], name as[имя], dr as [дата\_рождения] FROM таблица

Выражения в SELECT

SELECT

Фамилия, имя, Year(GetDate())-Year([дата\_рождения]) FROM таблица

Текстовые функции:

Right(строка, н) (справа выделяет н символов из строки)

Left(строка, н) (возвращает н символов строки слева)

Len(стрка) (возвращает длину строки)

Substring(строка, м, н) (возвращает н символов строки слева начиная с м-ого символа)

Rand() генерирует случайное число в интервале от 0 до 1 (0 включительно, 1 нет)

CAST(выражение AS тип) преобразует выражение от одного типа к другому

Виды операторов:

1. Арифметические +-\*/
2. Сравнения < > <= >= =
3. Логические AND OR NOT
4. Строковые +(конкатенация)
5. Вспомогательные IN Between Like

Оператор Like

Символ значение

% любая последовательность символов

\_ один символ

[ ] символ из списка

[^]

Задание 2

Вывести все записи из таблицы Студенты с одним полем вида: фамилия и о, имеющим название ФИО

SELECT Fam+’’+ LEFT(Name,1)+’’+ LEFT(Ot,1) AS [фио]

FROM Студенты

**Удаление всех повторяющихся полей из выборки**

SELECT DISTINCT список полей

FROM таблица

**Сортировка записей**

SELECT

FROM

ORDER BY(по возрастанию – ASC, по убыванию – DESC, если не писать, то по умолчанию ASC)

Пример соединения таблиц

SELECT таблица1.поле1 таблица2.поле1

FROM таблица1

INNER/LEFT/RIGHT JOIN выражение ON таблица2

!!!ЗАДАНИЕ на дом

!!!Фамилии студентов у которых нет оценок в таблице успеваемости

**Изменение данных в таблице**

UPDATE таблица

SET поле1=новое\_значение1…

Функции изменения регистра букв

UPPER(строка) – изменение на верхний регистр

LOWER(строка)-изменение на нижний регистр

**Удаление записей**

DELETE FROM таблица

WHERE условие отбора

Задание

Вывести фамилию группу оценку студентов, у которых оценка 2

Список отсортировать по группе, а внутри группы по фамилии

Задание 2

Всех студентов в таблице студенты перевести на курс старше

Групповые функции

Групповые функции работают с множеством строк и возвращают один результат на группу.

Множество строк может быть целой таблицей или таблицей, разбитой на группы.

Вывести минимальный номер, максимальный и кол-во студентов.

GROUP BY разбивает таблицу на группы, групповая функция возвращает 1 результат на группу

HAVING – условие отбора групповых данных.

Примечания

1. Список столбцов в group by обязателен
2. Использование псевдонимов недопустимо
3. Все столбцы которые входят в список select к которым не применяются групповые функции, должны быть указаны в group by

Задание напечатать список групп где девушек больше 5

Жизненный цикл БД

Жизненный цикл БД – это процесс проектирования, реализации и управления БД.

**АНАЛИЗ**(планирование разработки, определение требований, сбор и анализ требований)🡪**ПРОЕКТИРОВАНИЕ**(Концептуальная модель, логическая модель, физическая модель)🡪**Реализация**🡪**Тестирование**🡪**Эксплуатация и сопровождение**

**АНАЛИЗ**

Планирование разработки:

1. Объем работ
2. Требуемые ресурсы
3. Сроки разработки
4. Стоимость проекта

Время

Стоимость

Объем разработки

Качество

**Планирование разработки**

Решаются вопросы:

1. Технологической осуществимости
2. Операционной осуществимости
3. Экономической целесообразности

Технологическая осуществимость – есть ли соответствующее техническое и программное обеспечение для реализации проекта.

Операционная осуществимость – есть ли эксперты и разработчики, которые смогут принять участие в разработки проекта.

Экономическая целесообразность – окупится ли проект и принесет ли прибыль.

**Определение требований к системе**

Цели информационной системы:

1. Определение архитектуры(локальный/удаленный)
2. Определение требований к техническому и программному обеспечению
3. Информационные потребности отделов, количество пользователей

**Сбор и анализ требований**

Определение потребностей пользователей, на основании которых составляется спецификация требований пользователей.

После этапа анализа составляется техническое задание, которое подписывается обеими сторонами, и происходит переход к следующему этапу – проектирование БД.

**ПРОЕКТИРОВАНИЕ БД**

**Концептуальное проектирование**

Концептуальное проектирование БД – процесс создания модели, используемой на предприятии информации без физических аспектов ее представления.

Концептуальная модель

В результате изучения предметной области определяются информационные объекты, их атрибуты, связи между ними, ключевые поля.

ER-модель Чена

ER (entiny-relationship) сущность – связь

Эта модель была разработана Ченом в 1976 году с целью упрощения концептуального проектирования БД.

Элементы модели

Сущность 🡪 прямоугольник

Ключевые поля 🡪 овал(первичные ключи)

Связи 🡪 ромб (внутри ромба пишется глагол который характеризует связь)

Сущность – различимое множество объектов

**Логическое проектирование**

Процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов.

Логическая модель

Для построения логической модели БД используются средства автоматизации проектирования (CASE средства).

**Физическое проектирование**

Физическая модель

Задача физического этапа проектирования – выбор рациональной структуры хранения данных и методов доступа к ним.

Задание

На основе бланка заказа разработать концептуальную модель БД для фирмы, занимающейся автоматизированным обслуживанием заказов.

Пример

Заказ №777

|  |  |  |  |
| --- | --- | --- | --- |
| ДАТА  28.10.2014 | Поставщик №55 | ООО Салют, Ул Весенняя, д4 | Клиент №12 УниКит, Костомаровская набережная, 29 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер заказа | Название | Кол-во | Цена | Сумма |
| 1 | Стул | 100 | 500 | 50000 |
| 2 | Шкаф | 10 | 20000 | 200000 |
| 3 | Стол | 30 | 3000 | 90000 |

Определяемые атрибуты:

Клиенты(номер\_клиента, адрес, название)

Поставщик(номер\_пост, адрес)

Товары(номер\_твара, название, цена)

Заказы(номер\_заказа, номер\_товара, дата, кол-во, номер\_клиента, номер\_поставщика)

Таблица подробности\_о\_заказе(номер\_заказа, номер\_товара)

Доделать(продумать как в заказе указать несколько товаров)

клиенты

товары

заказы

поставщики

Номер\_тов

Номер\_кл

Номер\_заказа

Номер\_пост

доставляют

заказывают

входят

**РЕАЛИЗАЦИЯ**

Реализация – это разработка программы, позволяющей работать с БД.

**ТЕСТИРОВАНИЕ**

**ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ**

БД считается полностью разработанной и реализованной. Система наблюдается и поддерживается.

Задание: спроектировать бд для регистрации телефонов (мобильных, домашних, служебных) студентов колледжа, числа телефонов каждого вида произвольно.

Студенты(номер\_студента, фамилия\_студ, номер)

Телефон(номер\_студента, тип\_телефона, номер)

Виды\_телефонов(Вид\_телефона)

ДЗ –спроектировать бд, позволяющую вести учет книг в библиотеке колледжа.

Книги(Номер\_книги, Название\_книги, Наименование\_издательства, Год\_издания, Тематика, Стоимость)

Студенты(Номер\_студента, ФамилияИО, Группа\_студента)

Авторы(Номер\_автора, ФамилияИО\_автора)

Учет\_книг(Номер\_выдачи, Номер\_студента, Номер\_книги, Дата\_выдачи, Дата\_возврата)

Компания(в ней 4 отдела)

Отделы(1 отдел на 1 компанию, 1+ сотрудников)

Сотрудники(1 сотрудник работает только в 1 отделе, 0+иждивенцев, иметь или не иметь стаж работы)

Иждивенец(только к 1 сотруднику)

--------

Компания (номер\_компании, Наименование\_компании)

Отдел (Номер\_отдела, название\_отдела, номер\_компании)

Сотрудники(Номер\_сотрудника, ФамилияИО, номер\_отдела, номер\_компании)

Иждивенец (Номер\_иждивенца, номер\_сотрудника)

Трудовая\_книжка (номер\_сотрудника, стаж, номер\_компании, номер\_отдела)

ДЗ 11.12

Отделения (Номер отделения, Улица, Город, Штат, Почтовый индекс, Телефон, Номер менеджера)

Сотрудники (Табельный номер сотрудника, Имя, Должность, Зарплата, Номер отделения)

Жанр (Номер жанра, название жанра)

Видеокассеты (Номер по каталогу, Номер кассеты, название, Суточная плата за прокат, номер отделения)

Актёры и режиссёры (Номер участника, имя, должность)

Менеджеры (Табельный номер сотрудника, номер отдела)

Учёт проката (Номер договора, Номер клиента, Дата выдачи кассет, Дата возврата кассет)

Клиенты (Номер клиента, Имя, Фамилия, Адрес, Дата регистрации, номер отделения)

Прокат (Номер договора, Номер клиента, Номер кассеты, Номер по каталогу)

Роли в фильмах (Номер по каталогу, роль)

**Нормализация отношений**

Пример не нормализованной таблицы, поскольку в ней имеется излишнее дублирование данных

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Фамилия | Код специальности | Название специальности |
| 1 | Иванов | 09,02,03 | ПКС |
| 2 | Петров | 09,02,03 | ПКС |
| 3 | Сидоров | 09,02,02 | КС |
| 4 | Кузнецов | 09,02,02 | КС |

В ненормализованной БД присутствует излишнее дублирование, а также аномалии удаления и модификации.

Аномалии состоят в том, что невозможно что-либо сделать с 1 объектом не затронув другой объект.

Аномалия удаления – при удалении последнего студента со специальности удаляются сведения и сведения о специальности.

Аномалия вставки – невозможно добавить новую специальность без принятия на эту специальность хотя бы 1 студента.

Аномалия модификации – при изменении названия специальности существует вероятность ошибки и тогда может получиться что одному коду специальности соответствовать несколько названий и будет непонятно какое название правильное.

Нормализация отношений – приведение БД к эталонному виду, при котором отсутствует излишнее дублирование данных, устранены аномалии и уменьшены трудозатраты на ведение БД.

Нормализацию проводят путём приведения БД к нормальным формам.

Путь 1: архитектора.

Путь 2: разработчика.

1 нормальная форма – отношение называется приведённым к 1 нормальной форме если все его атрибуты просты далее неделимы.

Стр 490 конноли

2 нормальная форма – отношение находится во 2 нормальной форме если оно соответствует требованиям 1 нормальной формы и каждый не ключевой атрибут функционально зависит от первичного ключа.

Понятие функциональной зависимости атрибутов

Пусть в отношении R атрибут B функционально зависит от атрибута A если в каждый момент времени одному значению атрибута *А* соответствует только одно значение атрибута *В*. Функциональная зависимость графически отображается стрелкой:

А

В

Задание привести к 2 нормальной форме

Служащие

Номер работника

СЛУЖАЩИЕ

Номер служ

Фамилия

имя

отчество

ДЕТИ-----------------------

|

Номер ребенка -------\*

|

Номер служащего ---\*

|

Имя🡨---------------------

|

Дата рождения🡨-----

Архитектуры централизованных сетевых баз данных.

По технологиям обработки данных БД делятся на:

1. Централизованная БД хранится на одной машине компьютерной сети
2. Распределённая БД состоит из нескольких частей, возможно дублирующих друг друга, которые находятся на нескольких машинах вычислительной сети.

По способу доступа к данным БД делятся на:

1. С локальным доступом
2. С удалённым доступом

Системы централизованных БД с сетевым доступом предполагают различные архитектуры подобных систем:

1. Файл - сервер
2. Клиент - сервер

**Файл-сервер** – это выделенный сервер, предназначенный для выполнения файловых операций ввода-вывода и хранящий файлы любого типа.

Полностью скаченная БД обрабатывается средствами, имеющимися там в СУБД.

Недостатки:

* Высокий сетевой трафик (от сервера к клиенту);
* Низкая производительность;
* Слабая защита данных (только средствами ОС защита происходит);
* Пассивная роль сервера;

Достоинства:

* Доступ многих пользователей;
* Невысокая стоимость;
* Высокая скорость разработки;

**Клиент–сервер** - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами.

Клиент посылает запрос к серверу, сервер его обрабатывает и отправляет обратно результат.

Достоинства:

* Высокая производительность;
* Низкий трафик;
* Защита выше, чем у файл-сервер;
* Одновременный доступ многих пользователей;

Особенность – хранение используемых процедур и триггеров для реализации бизнес логики.

Триггер – вид хранимой процедуры, которая срабатывает при срабатывании одного из событий.

Недостатки:

* Высокая стоимость разработки;
* Медленная разработка (долго);
* Сервер может быть перегружен (высокая загрузка сервера);

Трёхзвенная архитектура используется с целью разгрузки сервера, то есть отводится дополнительная машина и на которой находится сервер приложений – клиентов, а на базовом сервере хранится БД и СУБД.

Вид:

Клиент -> сервер приложений -> сервер БД

Недостатки:

* Высокая стоимость
* Загрузка сервера

Достоинства:

* Высокая защита

**Подзапросы**

Пример запроса с подзапросом, в котором нужно вывести служащих оклад которых больше чем оклад у второго

SELECT Фамилия

FROM

WHERE Оклад >

(SELECT Оклад

FROM Служащие

WHERE Номер=2)

1. Подзапрос – это команда SELECT, включённая в описание другой команды
2. Подзапрос выполняется 1 раз до главного запроса
3. Результат подзапроса используется главным запросом

Подзапрос можно использовать с FROM WHERE и HAVING

Типы подзапросов:

1. Однострочный – подзапрос возвращает одну строку/значение (однострочные операторы =, >, <, <>, >=, <=)
2. Многострочный – возвращает несколько значений (используется с IN вместо =, >, <, <>, >=, <=)

Советы:

1. Использовать \* в списке полей SELECT по назначению
2. Используйте псевдонимы таблиц по назначению  
   SELECT stud.Фамилия   
   FROM Студенты stud я(stud – псевдоним таблицы)
3. Создавайте подзапросы в тех случаях, когда запрос основан на неизвестных значениях
4. Никогда не использовать \* в групповых запросах

**Управляющие конструкции Transact SQL**

Оператор объявления переменных

DECLARE

{@локальная\_переменная тип\_данных}

[,…n]

Set @локальная\_переменная=значение (присвоение значений)

SELECT @Локальная\_переменная = значение

BEGIN и END – операторные скобки

IF условие

{Выражение/блок выражений}

[ELSE

{Выражение/блок выражений}]

IF c EXISTS

EXISTS(подзапрос)

Возвращает true если в подзапросе есть хотя бы 1 запись, иначе возвращает false

CASE выражение

WHEN условное\_выражение THEN результирующее\_выражение

…

[ELSE…]

Пример c case

SELECT фамилия, брак, семейное\_положение=

CASE

WHEN брак <>1 THEN ‘не состоит в браке’

ELSE ‘состоит в браке’

END

FROM sort

Используя 2 формат case для мужчин вывести женат/не женат, а для женщины – замужем/не замужем

SELECT фамилия, брак, семейное положение=

Case

WHEN пол=’М’ AND брак=1 THEN ‘женат’

WHEN пол=’М’ AND брак<>1 THEN ‘холост’

WHEN =’Ж’ AND =1 THEN ‘замужем’

ELSE ‘свободна’

END

FROM sort

2 семестр

**Хранимые процедуры**

Хранимые процедуры – именованный набор команд transact sql, представляющий объект БД, который хранится и выполняется на сервере.

Выполнение хранимой процедуры:

1. Лексический анализатор выражений разбивает процедуру на отельные компоненты.
2. Сопоставление компонентов с объектами
3. Sysobjects – вносится запись о хранимой процедуры  
   syscomments – вносится текст процедуры
4. Создаётся план выполнения процедуры, и он записывается в sysprocedures
5. План считывается и оптимизируется при первом выполнении, а при повторном выполнении сразу выполняется.

Достоинства хранимых процедур: высокая производительность, уменьшение нагрузки на сеть, защита от sql-инъекций.

Типы хранимых процедур:

1. Системные (начинаются с префикса sp\_)
2. Пользовательские
3. Временные (имя начинается с # если локальные, и с ## если глобальные)

Создание хранимой процедуры

USE имя бд

IF EXISTS ( SELECT \* FROM SYSOBJECTS WHERE name = ‘имя’ AND type=’P’)

DROP PROC имя процедуры  
GO

CREATE PROC имя [@параметр тип\_данных

[OUTPUT][,…]] если выходной параметр

AS

BEGIN

---:операторы  
 END

GO вызов ХП

EXEC имя процедуры [список параметров]

1. ADO.NET
2. Entity Framework

Приложение

БД

ADO.NET

Connection

Command

DataReader

DataAdapter

DataTable

DataSet

Есть подключен

Нет подключения

Через объект connection идет подключение к БД, command выполняет команды на Transact-SQL, записи которые возвращает select помещаются в DataReader

Entity Framework (подходы)

1. DataBaseFirst
2. CodeFirst

Подключение модуля работы с sql на c#

USING SYSTEM.Data.SQLClient

Класс Connection

Важные свойства

Connection String

Параметры важные

Data Source – имя сервера

Initial Catalog – имя бд

Integrated Security

State (2 режима: open (соединение открыто), close(соединение закрыто) )

Методы:

Open и Close

Класс Command

Свойства

Command Text – текст SQL параметра

Connection – имя объекта connection

Важные команды:

ExecuteNonQuery – выполнение всех комманд кроме Select

ExecuteReader – выполнение select

DataReader

Метод Read

поля начинаются с нуля

Транзакции в SQL

Транзакция – группа команд, выполняемых единым блоком, по принципу: всё или ничего. Если выполняются все команды, происходит фиксация транзакции, в противном случае – откат.

Atomicity (неделимость) – транзакция неделима в том смысле, что представляет собой единое целое.

Consistency (согласованность) – после завершения транзакции информация в базе данных находится в согласованном состоянии

Isolation(изолированность) – параллельные транзакции выполняются изолированно друг от друга.

Durability(устойчивость) – результаты зафиксированной транзакции не могут быть потеряны ни при каких обстоятельствах.

Транзакции бывают **явные** и **автоматические** (неявные). Все SQL команды выполняются как неявные транзакции.

**Явные транзакции**

BEGIN TRAN[SACTION]

[имя\_транзакции|@имя\_переменной\_транзакции [WITH MARK ['описание\_транзакции’]]]

Фиксация транзакции

COMMIT [TRAN[SACTION] [имя\_транзакции|@имя\_переменной\_транзакции]]

Откат транзакции

ROLLBACK [TRAN[SACTION] [имя\_транзакции|@имя\_переменной транзакции]

Задание создать х.п. которая удаляет из таблицы назначения запись с заданным номером сотрудником, сделать это внутри транзакции с помощью параметра, управлять транзакции, при одном значении – откат, при другом – фиксацию.

Глобальная переменная @@TRANCOUNT содержит количество активных транзакций в системе.

Осуществить прием сотрудника на работу, используя транзакцию, при наличии вакансий

TRY – CATCH

отлавливает ошибку в try и перенаправляет ее в блок catch

--RAISERROR (‘Error raised in TRY block.’, --Message text. 16 – серьёзность, 1 – состояние)

Пример

CREATE PROC addPersonal @Surname nvarchar(30), @departmentCode INT, @codePosition INT, @addDate DATE

AS BEGIN

DECLARE @f INT, @maxPers INT, @working INT, @Nemp int

SET @f = 0

IF NOT EXISTS (SELECT \* FROM [Отделы] WHERE [код\_отдела] = @departmentCode)

BEGIN

RAISERROR(‘нет такого кода %d’, 16, 1, @departmentCode)

SET @f = 1

END

… точно так же с должностями

SET @maxPers = (SELECT Количество FROM Штат WHERE код\_должности = @codePosition AND код\_отдела = @departmentCode)

SET @working = (SELECT COUNT(номер) FROM Назначения WHERE код\_должности = @codePosition AND код\_отдела = @departmentCode AND дата\_увольнения IS NULL)

BEGIN TRAN

IF EXISTS (SELECT \* FROM сотрудники)

SELECT @nEpl = MAX(номер) FROM сотрудники

SET @nEpl = @nEpl +1

INSERT INTO сотрудники(номер, фамилия) VALUES (@nEpl, @Surname)

INSERT INTO назначения (номер, код\_должности, код\_отдела, дата\_приема) VALUES (@nEpl, @codePosition, @departmentCode, @addDate)

COMMIT

END

С сотрудниками колледжа

Создать хп в которой удаляется сотрудник с возможностью восстановления с помощью транзакций

Создать хп в которой перевести сотрудника с одной должности на другую используя транзакцию

Создать хп в которой прием сотрудника на должность директора не моложе 25 лет и не старше 50 используя транзакцию

С студентами колледжа

Создать хп в которой прием студента в группу если кол-во студентов не более 25

Создать хп в которой перевести студента из одной группы в другую если кол-во студентов не более 25 с возможностью восстановления

Используя таблицы студенты оценки и группы создать функцию которая возвращает качественную успеваемость по заданной группе

Создать пользовательскую функцию которая проверяет логин пользователя

Логин не должен повторяться 1 символ – буквы латинского алфавита, а после либо буквы либо цифры от 0 до 9 и не имеет знаки арифметических действий и тире

Триггеры

Триггеры – это вид хранимой процедуры, которая вызывается не явно, при выполнении какого-либо условия.

Триггер создаётся для определенной таблицы

Inserted

В таблице inserted сдержатся все строки, которые пользователь пытается вставить в таблицу

Deleted

В таблице Deleted не будет ни одной строки

После завершения триггера все строки из таблицы inserted переместятся в исходную таблицу

Команды update

При ее выполнении в таблице deleted находятся старые значения строк, которые будут удалены при успешном завершении триггера

Новые значения строк содержатся в таблице inserted

Эти строки добавятся в исходную таблицу после успешного выполнения триггера

С помощью треггеров достигаются следующие цели

Проверка корректности введенных данных и выполенение сложенных ограничений целостности данных, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности установленных таблиц

Накопление аудиторской информации посредством фиксации сведений о внесенных изменениях

Создание триггера

Create trigger [owner.]triiger\_name

On [owner.]table\_name

For {insert update delete}

[with encryption]

As sql\_statements

При создании триггеров надо помнить

1. Нельзя создавать триггеры для временных таблиц
2. Триггеры не могут возвращать результирующий набор
3. Триггеры следует использовать для поддержания целостности данных
4. В триггерах нельзя использовать команды create, drop, alter, truncate

Отмена действий триггера

Alter table имя\_таблицы disable trigger имя\_триггера

Возобновление действий триггера

Alter table имя\_таблицы enable trigger имя\_триггера

Проконтролировать увольнение сотрудника с использованием триггера

Дата увольнения не должна быть меньше даты приема на работу

Create trigger uvol

On назначения

For update , insert

As

If exists(select \* from inserted where date\_end<date\_b)

Begin

Raiserror (‘’)

Rollback transaction

End

Создать триггер запрещающий принимать на должность главного бухгалтера лиц моложе 25 лет проверить работу триггера

Создать триггер запрещающий исправлять оценки студентов на повышение баллов

