Добавление столбца с пронумерованными ячейками (ID)

ALTER TABLE [Жюри]

ADD IdNumber INT IDENTITY

Добавление первичного ключа

ALTER TABLE [Жюри]

ADD CONSTRAINT PK\_YourTable

PRIMARY KEY(IdNumber)

Изменение типа колонки

ALTER TABLE [Страны]

ALTER COLUMN [Код2] float not NULL

Связь таблиц:

alter table [Участники] WITH NOCHECK add constraint FKnbxg3ua7b8c5d53wps69q6jh

foreign key ([страна]) references [Страны]

Изменить название столбца:

exec sp\_rename '[Города].[1]', '[ID]'

Запросы:

SELECT Division, Location

FROM Org

WHERE Deptnumb =

(SELECT Dept FROM Staff

WHERE ID = 16)

--ALTER TABLE [Участники] ADD CONSTRAINT FK\_COUNTRY\_с FOREIGN KEY ([страна]) REFERENCES [Страны]([Код2]);

ALTER TABLE [Мероприятия]

ALTER COLUMN [ID] float not NULL

--select \* from [Города] where [City] is NULL

ALTER TABLE [Мероприятия]

ADD CONSTRAINT PK\_YourTableMer

PRIMARY KEY([ID])

--DELETE FROM [Города]

-- WHERE [City] is NULL

exec sp\_rename '[Мероприятия].[[ID]]]', 'ID'

--ALTER TABLE [Города] CHANGE [1] [ID] int;

alter table [Мероприятие] WITH NOCHECK add constraint FKnbxg3ua7b8c5d53wps69q6jh

foreign key ([страна]) references [Страны]

INSERT – вставка новых данных

UPDATE – обновление данных

DELETE – удаление данных

INSERT – вставка новых данных

INSERT [INTO] имя\_таблицы [(список\_столбцов)] VALUES (значение1, значение2, ... значениеN)

Вставка всех столбцов таблицы: можно скопировать все данные таблицы и вставить их в другую таблицу

INSERT INTO первая\_таблица SELECT \* FROM вторая\_таблица;

Вставка отдельных столбцов таблицы. Можно скопировать только те столбцы таблицы, которые необходимо вставить в другую таблицу

NSERT INTO первая\_таблица(имена\_столбцов1) SELECT имена\_столбцов2 FROM вторая\_таблица;

имена\_столбцов1: имена столбцов, разделенные запятой(,) для таблицы 1.

имена\_столбцов2: имена столбцов, разделенные запятой(,) для таблицы 2.

Копирование определенных строк из таблицы. Можно скопировать определенные строки из таблицы для последующей вставки в другую таблицу с помощью условия WHERE с оператором SELECT. В этом случае нужно использовать соответствующее условие в WHERE.

INSERT INTO таблица1 SELECT \* FROM таблица2 WHERE условие;

таблица1: имя первой таблицы.

таблица2: имя второй таблицы.

условие: условие для выбора строк.

UPDATE – обновление данных

UPDATE имя\_таблицы

SET столбец1 = значение1, столбец2 = значение2, ... столбецN = значениеN

[FROM выборка AS псевдоним\_выборки]

[WHERE условие\_обновления]

Например, увеличим у всех товаров цену на 5000:

UPDATE Products

SET Price = Price + 5000

Используем критерий, и изменим название производителя с "Samsung" на "Samsung Inc.":

UPDATE Products

SET Manufacturer = 'Samsung Inc.'

WHERE Manufacturer = 'Samsung'

Более сложный запрос - заменим у поля Manufacturer значение "Apple" на "Apple Inc." в первых 2 строках:

UPDATE Products

SET Manufacturer = 'Apple Inc.'

FROM

(SELECT TOP 2 FROM Products WHERE Manufacturer='Apple') AS Selected

WHERE Products.Id = Selected.Id

С помощью подзапроса после ключевого слова FROM производится выборка первых двух строк, в которых Manufacturer='Apple'. Для этой выборки будет определен псевдоним Selected. Псевдоним указывается после оператора AS.

Далее идет условие обновления Products.Id = Selected.Id. То есть фактически мы имеем дело с двумя таблицами - Products и Selected (которая является производной от Products). В Selected находится две первых строки, в которых Manufacturer='Apple'. В Products - вообще все строки. И обновление производится только для тех строк, которые есть в выборке Selected. То есть если в таблице Products десятки товаров с производителем Apple, то обновление коснется только двух первых из них.

DELETE – удаление данных

DELETE [FROM] имя\_таблицы

WHERE условие\_удаления

Например, удалим строки, у которых id равен 9:

DELETE Products

WHERE Id=9

Или удалим все товары, производителем которых является Xiaomi и которые имеют цену меньше 15000:

DELETE Products

WHERE Manufacturer='Xiaomi' AND Price < 15000

Более сложный пример - удалим первые два товара, у которых производитель - Apple:

DELETE Products FROM

(SELECT TOP 2 \* FROM Products

WHERE Manufacturer='Apple') AS Selected

WHERE Products.Id = Selected.Id

После первого оператора FROM идет выборка двух строк из таблицы Products. Этой выборке назначается псевдоним Selected с помощью оператора AS. Далее устанавливаем условие, что если Id в таблице Products имеет то же значение, что и Id в выборке Selected, то строка удаляется.

SELECT – выборка данных

Базовый синтаксис команды SELECT выглядит следующим образом:

SELECT [DISTINCT] список\_столбцов или \*FROM источник WHERE фильтр ORDER BY выражение\_сортировки

ALTER TABLE имя\_таблицы ADD перечень\_полей\_с\_характеристиками

– позволяет добавить новые поля в таблицу;

ALTER TABLE имя\_таблицы DROP COLUMN перечень\_полей

– позволяет удалить поля из таблицы;

ALTER TABLE имя\_таблицы ADD CONSTRAINT имя\_ограничения

FOREIGN KEY(поля) REFERENCES таблица\_справочник(поля)

– позволяет определить связь между таблицей и таблицей справочником

Добавление свойства IDENTITY к полю

– позволяет сделать это поле автоматически заполняемым (полем-счетчиком) для таблицы;

Прочие ограничения – UNIQUE, DEFAULT, CHECK

При помощи ограничения UNIQUE можно сказать,

что значения для каждой строки в данном поле или в наборе полей должно быть уникальным.

CREATE TABLE Production.TransactionHistoryArchive4

(

TransactionID int NOT NULL,

CONSTRAINT AK\_TransactionID UNIQUE(TransactionID)

)

ALTER TABLE Employees ADD CONSTRAINT UQ\_Employees\_Email UNIQUE(Email)

ALTER TABLE имя\_таблицы ADD CONSTRAINT имя\_ограничения UNIQUE(поле1,поле2,…)

При помощи добавления к полю ограничения DEFAULT мы можем задать значение по умолчанию,

которое будет подставляться в случае,

если при вставке новой записи данное поле не будет перечислено в списке

полей команды INSERT.

ALTER TABLE Employees ADD HireDate date NOT NULL DEFAULT SYSDATETIME()

ALTER TABLE Employees ADD DEFAULT SYSDATETIME() FOR HireDate

ALTER TABLE Employees ADD CONSTRAINT DF\_Employees\_HireDate DEFAULT SYSDATETIME() FOR HireDate

Проверочное ограничение CHECK используется в том случае,

когда необходимо осуществить проверку вставляемых в поле значений.

Например, наложим данное ограничение на поле табельный номер,

которое у нас является идентификатором сотрудника (ID).

При помощи данного ограничения скажем, что табельные номера должны иметь значение

от 1000 до 1999:

ALTER TABLE Employees ADD CONSTRAINT CK\_Employees\_ID CHECK(ID BETWEEN 1000 AND 1999)

Можно так же создать ограничения UNIQUE и CHECK без указания имени:

ALTER TABLE Employees ADD UNIQUE(Email) ALTER TABLE Employees

ADD CHECK(ID BETWEEN 1000 AND 1999)

Все виды ограничений, которые создаются командой вида

«ALTER TABLE имя\_таблицы ADD CONSTRAINT имя\_ограничения …»:

PRIMARY KEY – первичный ключ;

FOREIGN KEY – настройка связей и контроль ссылочной целостности данных;

UNIQUE – позволяет создать уникальность;

CHECK – позволяет осуществлять корректность введенных данных;

DEFAULT – позволяет задать значение по умолчанию;

Все ограничения можно удалить, используя команду вида

«ALTER TABLE имя\_таблицы DROP CONSTRAINT имя\_ограничения».

Кластерный (CLUSTERED) и некластерный (NONCLUSTERED) индекс.

Создание самостоятельных индексов.

Под самостоятельностью здесь имеются в виду индексы,

которые создаются не для ограничения PRIMARY KEY или UNIQUE.

Индексы по полю или полям можно создавать следующей командой:

CREATE INDEX IDX\_Employees\_Name ON Employees(Name)

Так же можно указать опции CLUSTERED, NONCLUSTERED, UNIQUE,

а так же можно указать направление сортировки каждого отдельного поля ASC (по умолчанию) или DESC:

CREATE UNIQUE NONCLUSTERED INDEX UQ\_Employees\_EmailDesc ON Employees(Email DESC)

При создании некластерного индекса опцию NONCLUSTERED можно отпустить,

т.к. она подразумевается по умолчанию, здесь она показана просто,

чтобы указать позицию опции CLUSTERED или NONCLUSTERED в команде.

Удалить индекс можно следующей командой:

DROP INDEX IDX\_Employees\_Name ON Employees

Простые индексы так же, как и ограничения,

можно создать в контексте команды CREATE TABLE.

Создание таблицы со всеми созданными ограничениями и индексами одной командой CREATE TABLE:

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30),

Birthday date,

Email nvarchar(30),

PositionID int,

DepartmentID int,

HireDate date NOT NULL CONSTRAINT DF\_Employees\_HireDate DEFAULT SYSDATETIME(),

ManagerID int,

CONSTRAINT PK\_Employees PRIMARY KEY (ID),

CONSTRAINT FK\_Employees\_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES Departments(ID),

CONSTRAINT FK\_Employees\_PositionID FOREIGN KEY(PositionID) REFERENCES Positions(ID),

CONSTRAINT FK\_Employees\_ManagerID FOREIGN KEY (ManagerID) REFERENCES Employees(ID),

CONSTRAINT UQ\_Employees\_Email UNIQUE(Email),

CONSTRAINT CK\_Employees\_ID CHECK(ID BETWEEN 1000 AND 1999),

INDEX IDX\_Employees\_Name(Name)

)

В итоге:

Индексы могут повысить скорость выборки данных (SELECT),

но индексы уменьшают скорость модификации данных таблицы,

т.к. после каждой модификации системе будет необходимо перестроить все индексы

для конкретной таблицы.