SQL RELATIONSHIPS

**One to one relationship:**

**1.create** two **tables**, **then** define a **simple** **primary** **foreign** **key** relationship **between** them, **and** **set** the **foreign** **key** **column** **to** be **unique**.

**CREATE** **TABLE** Employee (

ID **int** **PRIMARY** **KEY**,

**Name** **VARCHAR**(50)

);

**drop** **table** Employee ;

**CREATE** **TABLE** Salary (

EmployeeID **int** **UNIQUE** **NOT** **NULL**,

SalaryAmount **int**

);

**drop** **table** salary ;

**ALTER** **TABLE** Salary

**ADD** **CONSTRAINT** FK\_Salary\_Employee **FOREIGN** **KEY** (EmployeeID)

**REFERENCES** Employee (ID);

**One to many relationship:**

Step 1:

**Create** two **tables** (**table** 1 **and** **table** 2) **with** their own **primary** keys.

**CREATE** **TABLE** city (

city\_id **int** **PRIMARY** **KEY** ,

city **varchar**(50) **NOT** **NULL**,

country\_id **int** **NOT** **NULL**

);

**drop** **table** city;

**CREATE** **TABLE** country (

country\_id **int** **primary** **KEY**,

country **varchar**(50) **NOT** **NULL**

);

**drop** **table** country;

step 2:

**Add** a **foreign** **key** **on** a **column** **in** **table** 1 based **on** the **primary** **key** **of** **table** 2.

This will mean that **table** 1 can have one **or** **more** records related **to** a single record **in** **table** 2.

**ALTER** **TABLE** city

**ADD** **FOREIGN** **KEY** (country\_id) **REFERENCES** country (country\_id);

--Many-to-one relationship

There **is** **not** much difference **between** one-**to**-many **and** many-**to**-one relationships. It’s just a matter **of** focus.

**If** one school **class** can consist **of** several pupils **then**, **class** **to** pupil **is** a one-**to**-many relationship (one **class** consists **of** many pupils),

**while** pupil **to** **class** relationship **is** many-**to**-one (many pupils study **in** one **class**).

**Self-referencing relationship:**

Step 1:

**First**, we **create** the two **tables**

**CREATE** **TABLE** films (

film\_id **INT** **PRIMARY** **KEY**,

title **VARCHAR**(50),

director **VARCHAR**(50),

year\_released **INT**

);

**CREATE** **TABLE** category (

category\_id **INT** **PRIMARY** **KEY**

,**name** **VARCHAR**(50)

);

Step2: we **create** a junction **table** film\_category that will **map**

these two **tables** together **by** **referencing** the **primary** keys **of** **both** **tables**.

--bridge table/linking table

**CREATE** **TABLE** film\_category (

film\_id **INT**,

category\_id **INT**,

**CONSTRAINT** film\_cat\_pk **PRIMARY** **KEY** (film\_id, category\_id),

**CONSTRAINT** FK\_film

**FOREIGN** **KEY** (film\_id) **REFERENCES** films (film\_id),

**CONSTRAINT** FK\_category

**FOREIGN** **KEY** (category\_id) **REFERENCES** category (category\_id)

);

**drop** **table** film\_category ;

**Self-referencing relationship:**

**CREATE** **TABLE** Staff (

StaffId **int** **PRIMARY** **KEY**,

FirstName **varchar**(50) **NOT** **NULL**,

LastName **varchar**(50) **NOT** **NULL**,

Email **varchar**(255) **NOT** **NULL**,

Phone **varchar**(25) **NULL**,

Active **smallint** **NOT** **NULL**,

StoreId **int** **NOT** **NULL**,

ManagerId **int** **NULL**,

**UNIQUE** (Email)

);

**ALTER** **TABLE** Staff

**ADD** **FOREIGN** **KEY** (ManagerId) **REFERENCES** Staff (StaffId);