**Create a table Key in MySQL…**

**Payment Entities -**

Invoices, invoice line items, surcharges, payment transactions, billing debit/credits card, debit/credit card type, currencies, booking.com partners, partners bank account details, commissions

**Payments Primary Keys –**

Every payment entity uses an auto incrementing INT(11) id for a primary key that corresponds to the entity name. For example, every invoice row will have an invoice\_id that enforces uniqueness.

The only exception to this is the billing debit/credits card entity which uses a foreign key as its primary key. That is a billing debit/credit card row is uniquely identified by the payment transaction it was used for, meaning that the billing debit/credits card entity is a weak entity.

**Information on Foreign Keys-**

*Bookings*

Every booking must have a customer and a customer can make zero to many booking(s) resulting in the customer\_id foreign key in the booking table.

*Booked Rooms*

A booking has one to many booked rooms and a room has been booked zero to many times (or a booking has one to many rooms and a room has zero to many bookings). This relationship results in the booked\_rooms table that has two foreign keys, a booking\_id from the booking table and a room\_id from the room table.

*Invoices*

A booking has one to many invoices (e.g. one invoice could be for a payment and another could be for a refund for a booking). This relationship results in the invoice table having a booking\_id foreign key.

An invoice must have a currency thus the invoice table has a currency\_id foreign key.

An invoice must have a billing address thus the invoice table has a billing\_address foreign key.

An invoice can be made out to either a customer or a booking.com partner (in the case of charges for booking.com commissions). Therefore the invoice table has customer\_id and booking.com\_partner\_id foreign keys.

*Invoice line items*

An invoice has one to many line items (e.g. a booking has two booked rooms and one parking spot, there will be a line for each of these and another line for vat and another line for any city tax payable, these lines will create the total invoice amount). This relationship results in the invoice line item table having an invoice\_id foreign key.

Each invoice line item can have one surcharge (or this can be null) for instance if a hotel were to charge for parking, child care or the mini bar etc. This relationship results in the invoice line item table having a surcharge\_id foreign key. ).

An invoice line item must correspond to a booked room and a booked room can have many invoice line items. This relationship results in the invoice line item table having a booking\_id foreign key and a booked\_room\_id.

*Payment transactions*

An invoice has zero to many payment transactions and each payment transaction has a transaction type (e.g. an online credit/debit card payment, a Paypal payment, a payment at the accommodation or a refund etc.). These relationships result in the payment transactions table having an invoice\_id foreign key and a transaction\_type\_id foreign key.

A payment transaction has to be from or to a customer (or a booking.com partner…). This relationship results in a customer\_id foreign key in the payment transaction table (and a booking.com\_partner\_id foreign key).

A payment transaction must have a currency. This relationship results in the payment transaction table having a currency\_id foreign key.

*Booking.com partners*

Each booking.com partner has one bank account to which payments can be made. This relationship results in the booking.com partners table having a partners\_bank\_account\_id foreign key.

*Billing debit/credit card*

A billing debit/credit card has one payment transaction (this information is stored, passed to the payment provider and then deleted after 10 days). This relationship results in the Billing debit/credit card table having a transaction\_id foreign key.

A billing debit/credit card has a debit/credit card type, thus the billing debit/credit card table has a billing debit/credit\_card\_type\_id.

A billing debit/credit card has a stored debit/credit card id if a customer has opted to use a stored debit/credit card (i.e. autofill the billing credit/debit card section). This relationship results in a stored\_credit/debit\_card\_id in the billing debit/credit card table.

*Stored debit/credit cards*

A customer can have zero or many stored credit/debit cards, thus the stored debit/credit cards table has a customer\_id foreign key.

A stored debit/credit card has a debit/credit card type, thus the stored debit/credit cards table has a thus the stored debit/credit cards table has a debit/credit\_card\_type\_id foreign key.

Each invoice has one billing address

A billing address can have one customer address id (if a customer has stored an address then they can autofill the billing address).

Each invoice can be made out to one customer (or customer\_id can be null) or it can have one booking.com partner (or booking.com\_partner\_id can be null)

If an invoice is payable to a booking.com partner then a commission will be taken from the payment, each booking.com partner can have zero or many commissions.