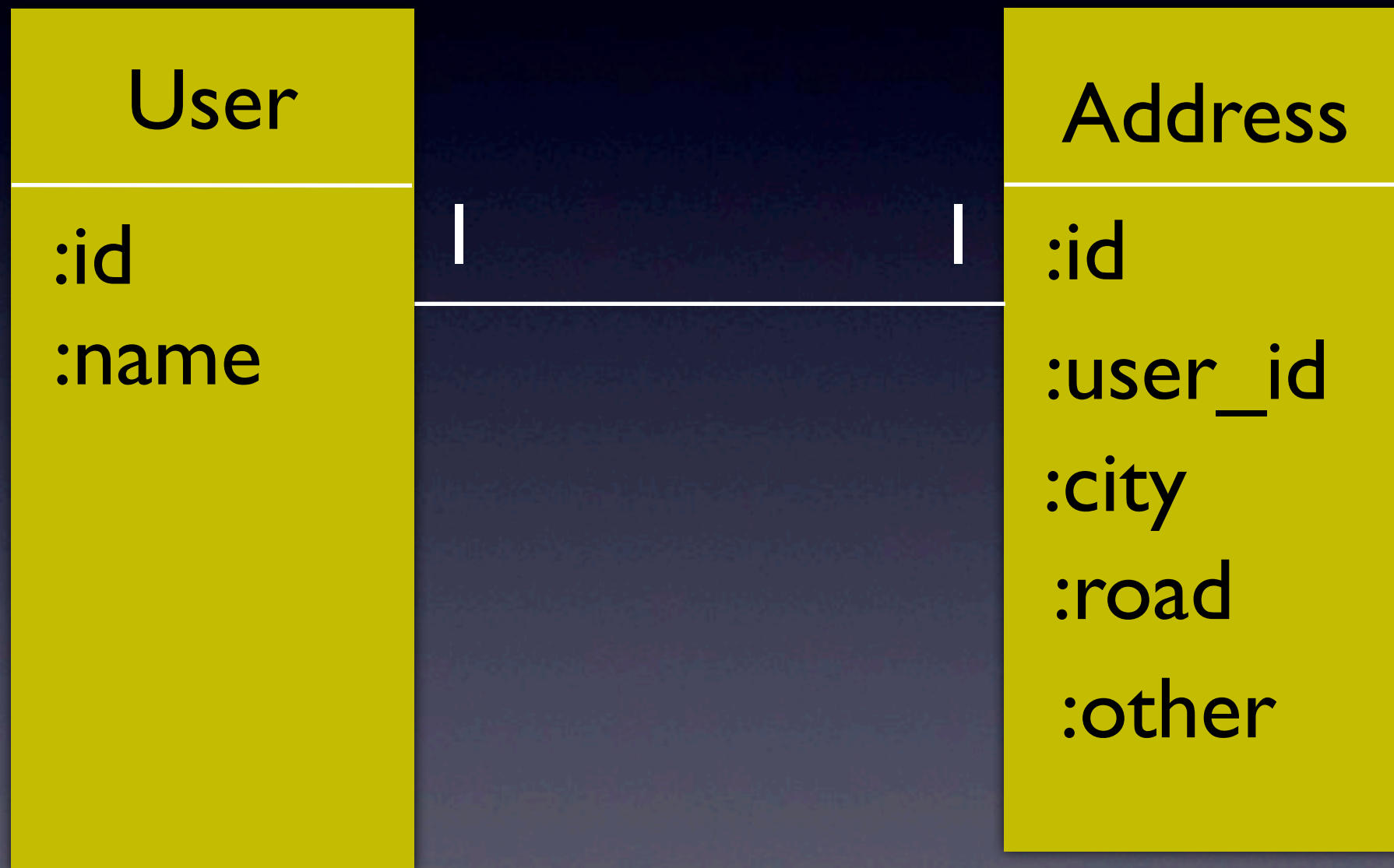


# Model Association

# Association

- ActiveRecord可以用Associations來定義資料表之間的關聯性。
- 資料表之間的關聯性共有三種：
  - one-to-one
  - one-to-many
  - many-to-many

# One-to-One (I)



# One-to-One (II)

```
class User < ActiveRecord::Base
  has_one :address
  #...
end
```

- 預設是使用Address這個model。
- address為單數。



# One-to-One (III)

```
class Address < ActiveRecord::Base
  belongs_to :user
  #...
end
```

- 預設是使用User這個model。
- 預設有user\_id當做foreign key。
- user為單數。

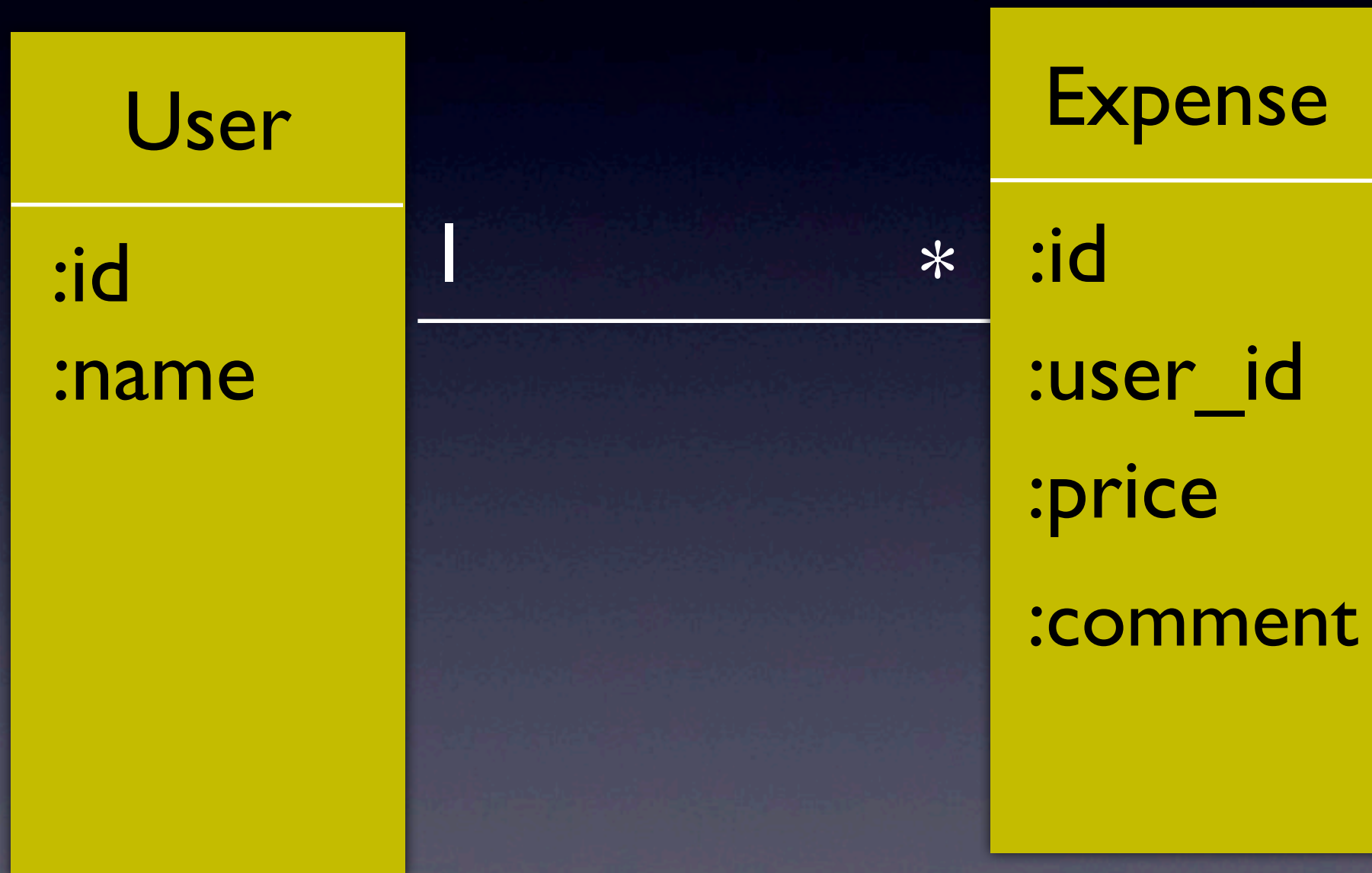
# 小練習

- 建立一個app，來實作User - Address的關係。
- 利用Rails console來檢驗是否正確。

```
u = User.create(:name => 'weijen')  
addr = u.build_address(:city=>"HsinChu",  
                        :road => "Zhonghua",  
                        :other => "No.53")
```

```
addr.save  
p addr  
p addr.user
```

# One-to-Many (I)



# One-to-Many (II)

```
class User < ActiveRecord::Base
  has_many :expenses
  #...
end
```

- 預設使用Expense這個model。
- expenses為複數。



# One-to-Many (III)

```
class Expense < ActiveRecord::Base
  belongs_to :user
  #...
end
```

- 預設是使用User這個model。
- 預設有user\_id當做foreign key。
- user為單數。

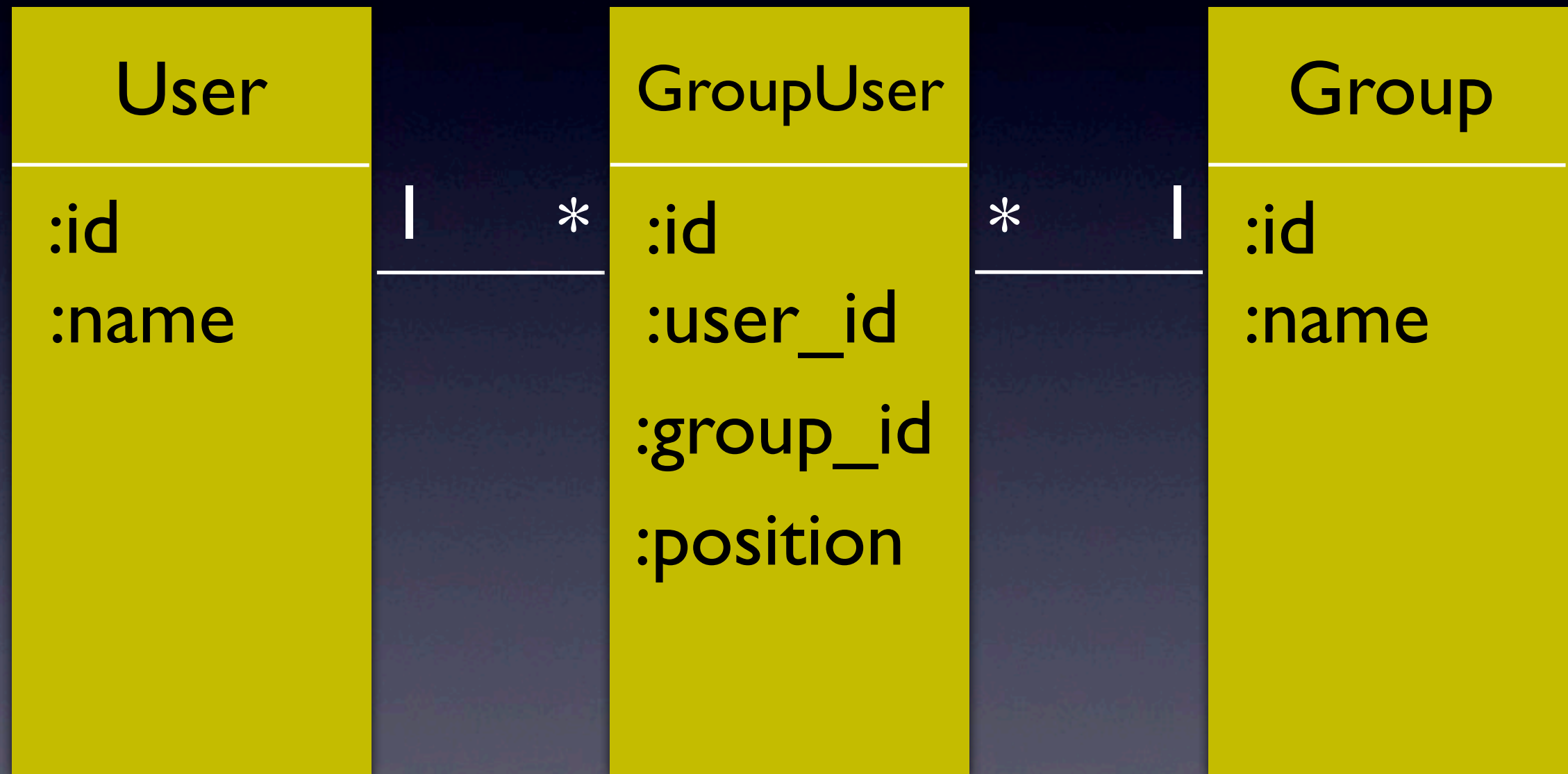
# 小練習

- 實作User - Expense的關係。
- 利用Rails console來檢驗是否正確。

```
u = User.first
expense = u.expenses.new(price: 100,
                          comment: "test comment")

expense.save
p expense
p expense.user
```

# Many-to-Many (I)



# Many-to-Many (II)

```
class Group < ActiveRecord::Base
  has_many :group_users
  has_many :users, :through
    => :group_users
end
class User < ActiveRecord::Base
  has_many :group_users
  has_many :groups, :through => :group_users
end
```

- 透過GroupUser來使User跟Group有many-to-many的關係



# Many-to-Many (III)

```
class GroupUser < ActiveRecord::Base
  belongs_to :group
  belongs_to :user
end
```

- 同時屬於Group 與 User model
- 預設有group\_id與user\_id
- 都是使用單數

# 小練習 (I)

- 實作Group-GroupUser-User的關係。
- 利用Rails console來檢驗是否正確。

```
u = User.first
```

```
g1 = Group.create(:name => "group1")
```

```
u.groups < g1
```

```
g2 = u.groups.create(:name => "group2")
```

# 參數 (I)

- `class_name`: 變更關聯的類別名稱
- `foreign_key`: 變更Foreign Key的欄位名稱

```
class Group < ActiveRecord::Base
  belongs_to :owner,
             :class_name => "User",
             :foreign_key => "owner_id"
end
```



# 參數 (II)

- `has_many` 可以利用Proc來指定順序等。

```
class User < ActiveRecord::Base
  has_many :groups,
    -> { order("id DESC") }
end
```



# 參數 (III)

- **dependent**：當物件刪除時，也會順便刪除它的has\_many物件。
- 有三種不同的刪除方式：
  - **:destroy**：會執行destroy callback
  - **:delete**：不會執行destroy callback
  - **:nullify**：這是預設值，不會幫忙刪除

# 參數 (IV)

```
class User < ActiveRecord::Base
  has_many :expenses, dependent: :destroy
  #...
end
```