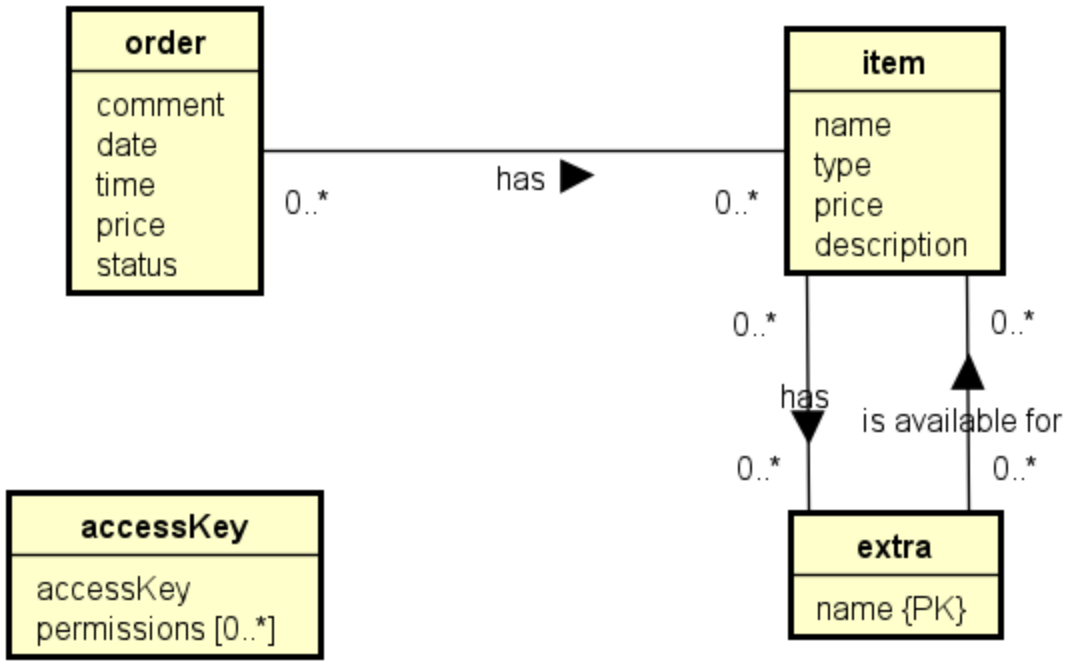
DATABASE DESIGN DOCUMENTATION

ER DIAGRAM



ORDER: An order is, in essence, composed of many items. It has a date and time in which it was made, and the price will be calculated inside the application and sent to the database directly. It also has a status, which can be “pending” or “completed”.

ITEM: An item has a name. It can be of different types, such as “coffee”, “tea”, etc. to facilitate sorting of items within the application. It has a price and may have a description.

EXTRA: Many types of extras exist. They are for free, so they don’t have a price. They may be added to an item in an order. The same extra might not be available for all items, e.g., it should not be possible to add honey to Pringles, but it should be available for tea, so there is a list of what extras are available for each item.

ACCESSKEY: In the application, not all users should have the same permissions (e.g., a barista’s only access is to mark orders as completed, not to accept an unpaid order). For this purpose, the employees have access Keys with different permissions.

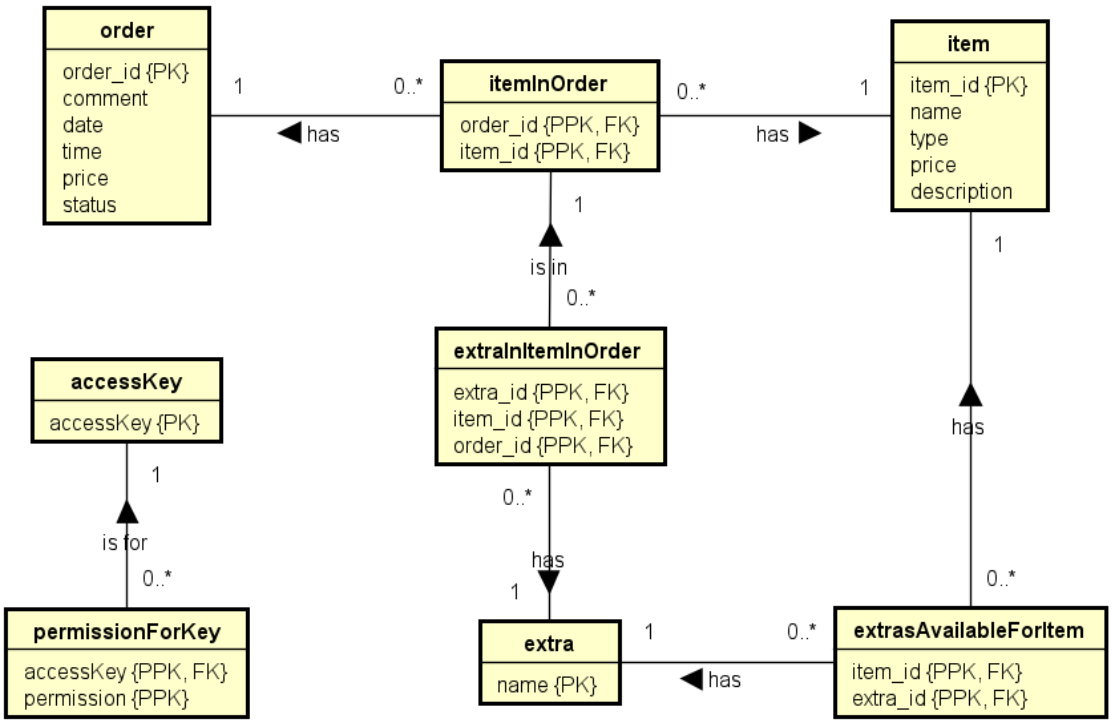
RELATIONAL SCHEMA

Order(order_id, comment, date, time, price, status) PK: order_id	**Item(item_id, name, type, price, description)** PK: item_id
Extra(name) PK: name	**AccessKey(accessKey)** PK: accessKey
ExtrasAvailableForItem(item_id, extra_id) PK: item_id, extra_id FK: item_id REF Item(item_id) FK: extra_id REF Extra(extra_id)	**ItemInOrder(order_id, item_id)** PK: order_id, item_id FK: order_id REF Order(order_id) FK: item_id REF Item(item_id)
ExtraInItemInOrder(order_id, item_id, extra_id) PK: order_id, item_id, extra_id FK: extra_id REF Extra (extra_id) FK: order_id, item_id REF ItemInOrder(order_id, item_id)	**PermissionForKey(accessKey, permission)** PK: accessKey, permission FK: accessKey REF AccessKey(accessKey)

After following the mapping steps, it was possible to come up with the relation schema found above.

1) **Strong entities:** The entities that do not depend on others to exist were mapped. It was necessary to introduce surrogate keys for some relations (order, item).
2) **Weak entities:** Not relevant
3) **One to many:** Not relevant
4) **One to one:** Not relevant
5) **1..1 to 1..1:** Not relevant
6) **1..1 to 0..1:** Not relevant
7) **0..1 to 0..1:** Not relevant
8) **Recursive relationships:** Not relevant
9) **Inheritance:** Not relevant
10) **Aggregation & Composition:** Not relevant
11) **Many to many:** Many to many relationships result in “join-relations”. The ER diagram featured many of these:
    a. **Order & Item -> ItemInOrder:** This relation makes it possible for an order to have multiple items.
    b. **Extra & Item -> ExtrasAvailableForItem:** This relation represents what extras are available for what items.
    c. **Item & Extra -> ExtraInItemInOrder:** This relation makes it possible for someone to add an extra to a certain item in their order. It references another join-relation.
12) **Complex relationships:** Not relevant
13) **Multivalued attributes:** Access Key & Permission [0..*] -> PermissionForKey: This relation makes it possible for multiple permissions to be granted to the same access key.

GLOBAL RELATIONS DIAGRAM



A visual representation of the relational schema can be expressed as the above Global Relations Diagram.