kokchun giang

going from a conceptual business-centric modeling to **logical** and **physical** modeling that are more technical

the **data modeling** journey for transactional data

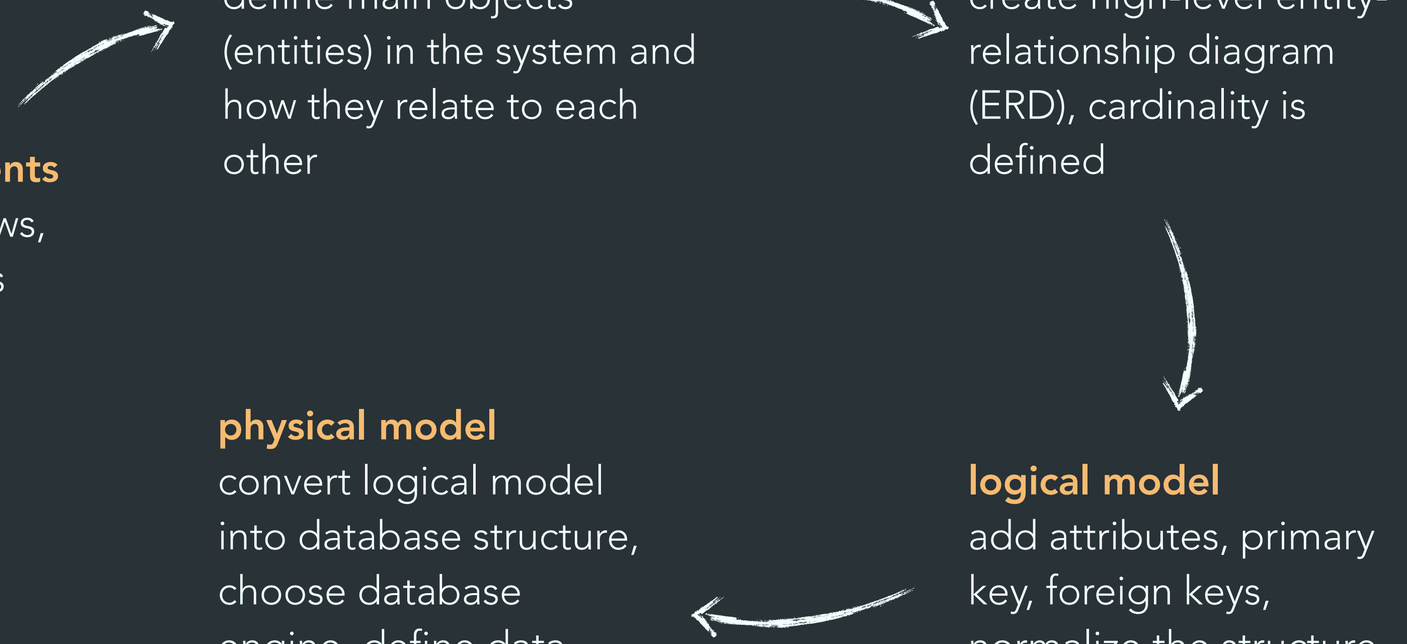
business requirements
stakeholder interviews, identify key business processes

entities & relationships
define main objects (entities) in the system and how they relate to each other

conceptual model
create high-level entity-relationship diagram (ERD), cardinality is defined

logical model
add attributes, primary key, foreign keys, normalize the structure

physical model
convert logical model into database structure, choose database engine, define data types, constraints, …



the **data modeling** journey for transactional data

business requirements
stakeholder interviews, identify key business processes

entities & relationships
define main objects (entities) in the system and how they relate to each other

conceptual model
create high-level entity-relationship diagram (ERD), cardinality is defined

logical model
add attributes, primary key, foreign keys, normalize the structure

physical model
convert logical model into database structure, choose database engine, define data types, constraints, …

remember the **business requirements** for ezecream

customers should be able to browse and order ice cream flavors online

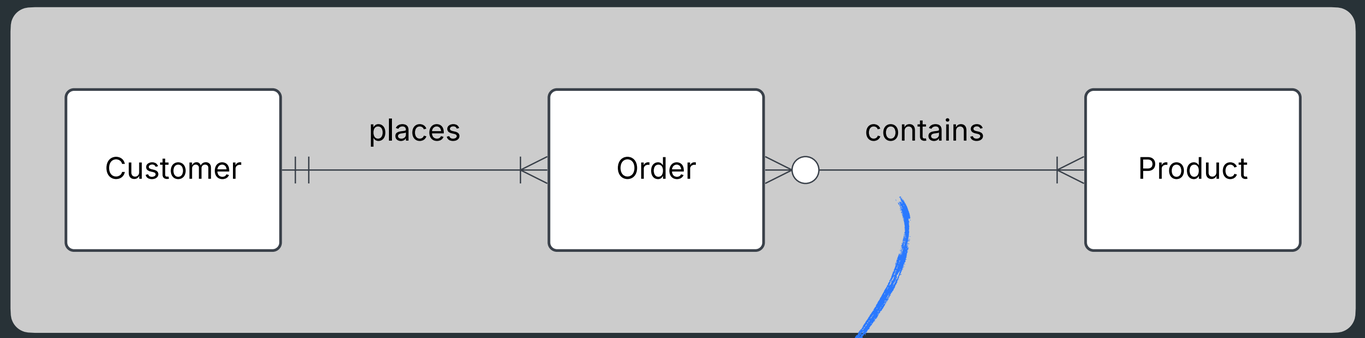
each order should contain one or more ice cream flavors

the system should store order details, including order date and total price

customers should provide their name, contact details, and delivery address

each ice cream flavor should have a name, price, and availability status

a **conceptual ERD** for ezecream using crows foot notation



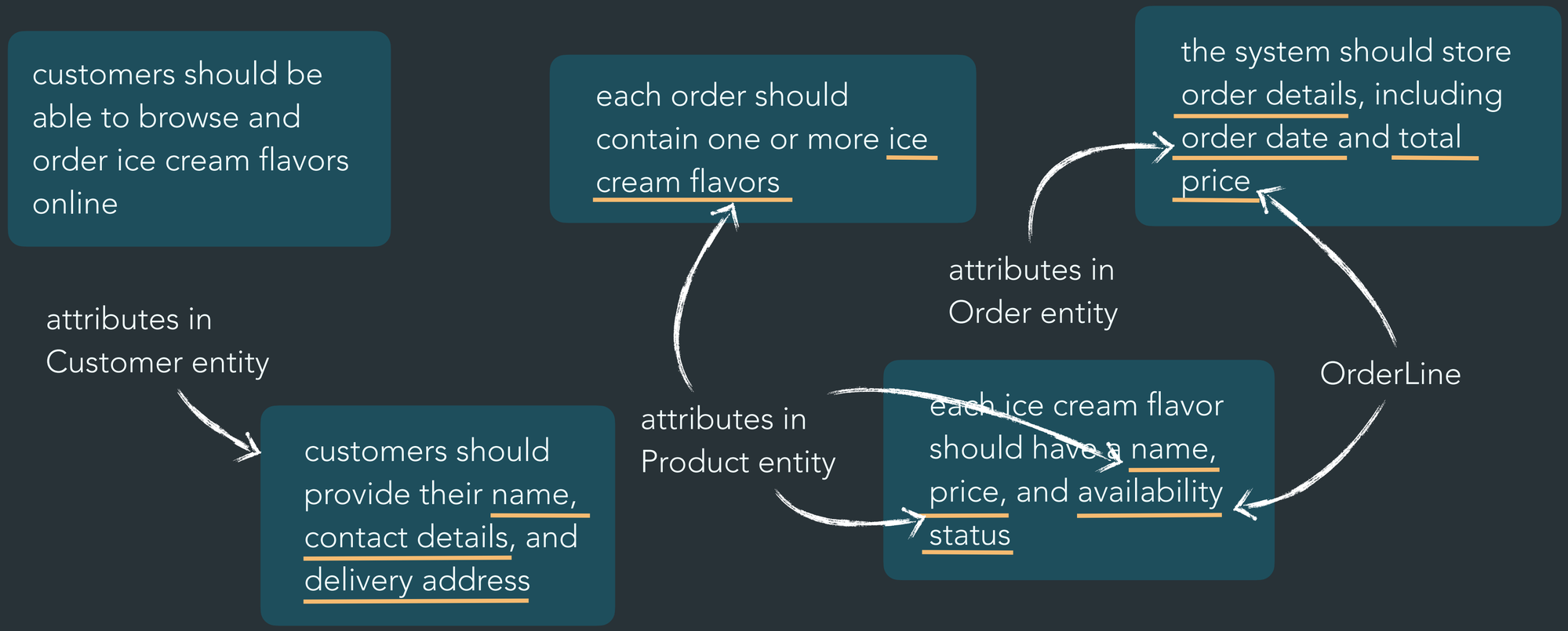
many-to-many relationship, can’t be implemented directly

introduce a **composite entity** OrderItem in between a many-to-many

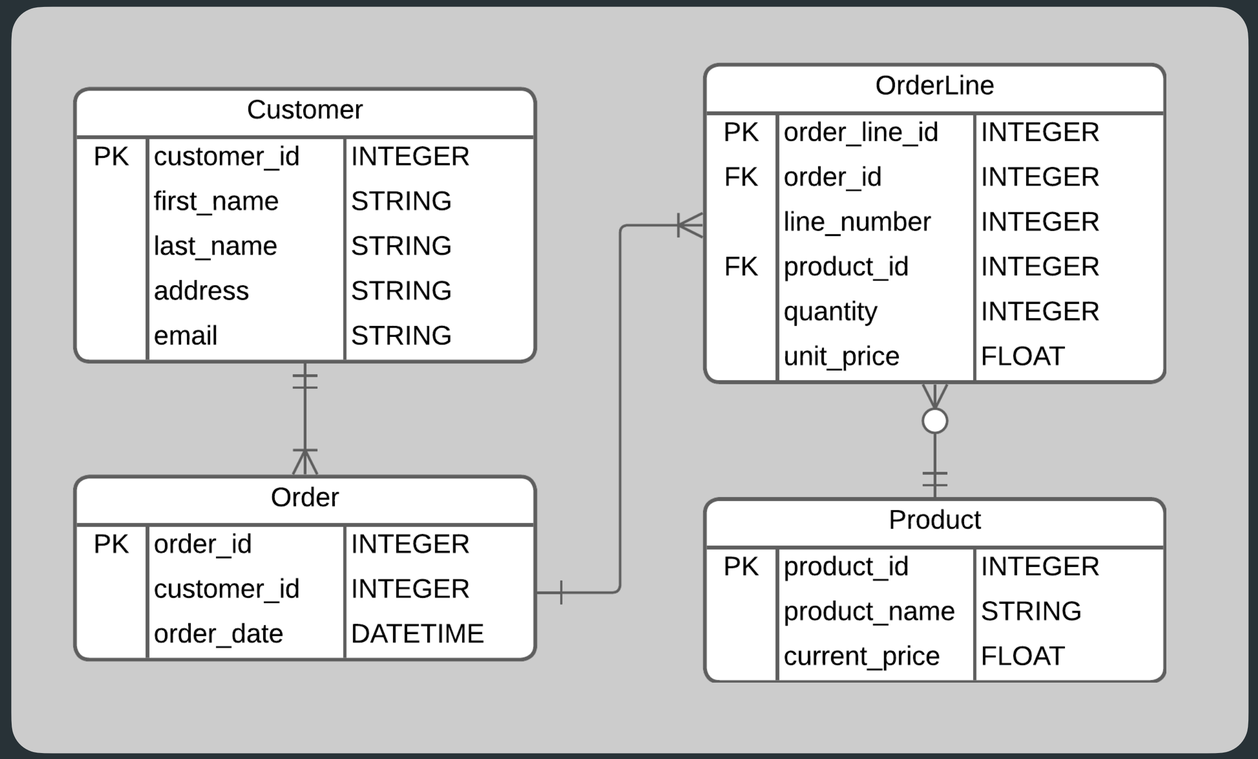
replaces a many-to-many relationship with two one-to-many relationships

it acts as a bridge table with foreign keys that references primary keys in the related tables

identify the **entities & relationships** from the requirements



the **logical data model** for ezecream



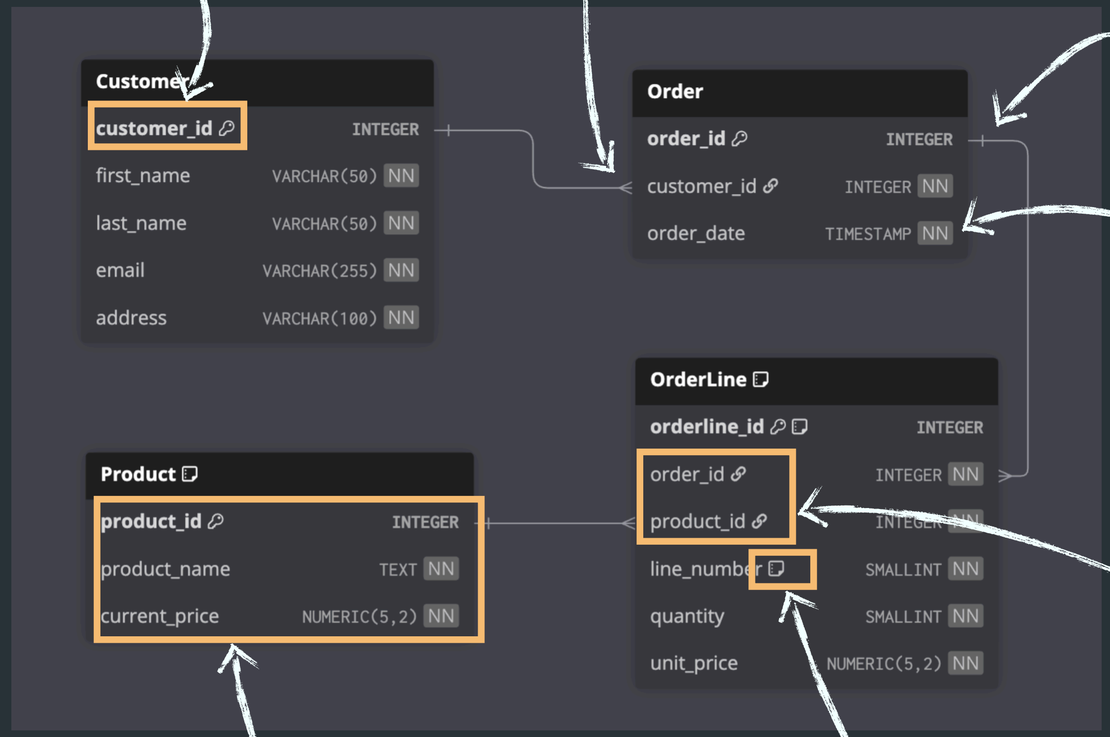
difference from conceptual model is that logical model also contains **attributes, keys & preliminary data types**

break up many-to-many to composite entity with two one-to-many

removed relationship labels

this makes the model more **specialized** while conceptual model reflects the real world more generally

the **physical data model** for ezecream using dbdiagram



primary key

not specified from zero
or from one in many side

this is one and only
one in dbdiagram

not null

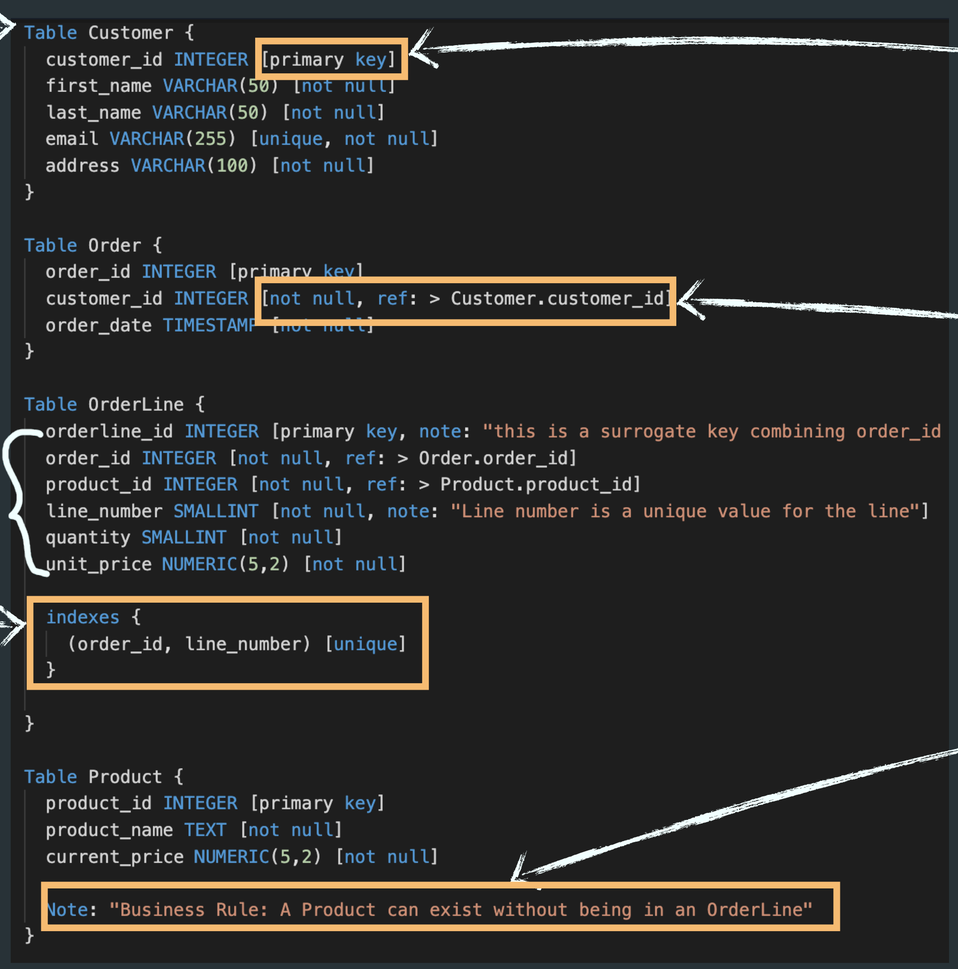
all non-null fields must be
entered to insert one record in
the database table

foreign key
reference to
primary key

attributes and data types
(specific to postgres)

notes that will show
when hovered

database modeling language **dbml** is used in dbdiagram to do physical data modeling



Customer table

primary key constraint

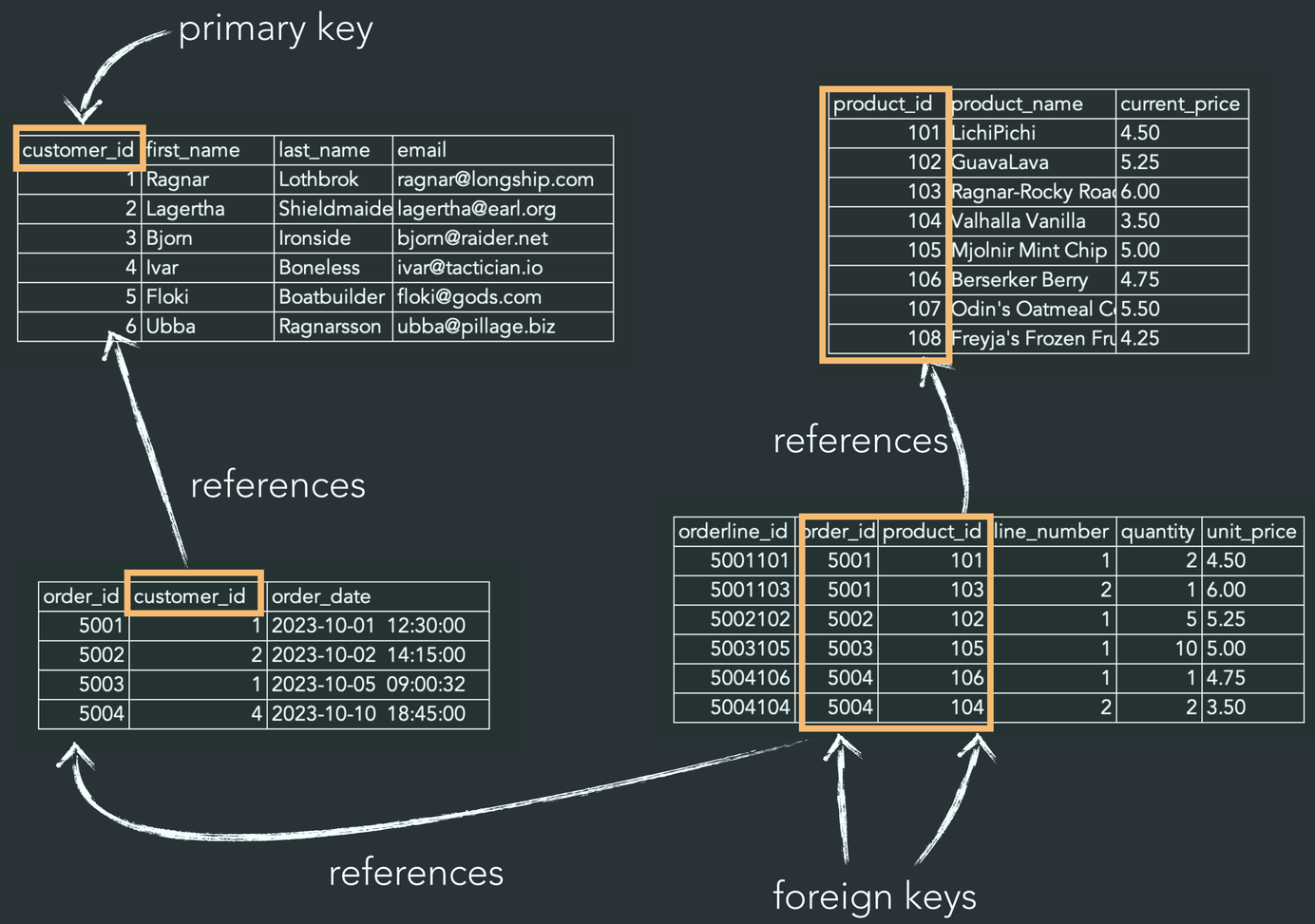
not null constraint and foreign key referencing primary keys in related table

attributes or fields

stop duplicates

notes shown when hover the diagram

the **actual tables** could look like this in the end



foreign keys in the bridge table references primary keys in the related tables

can you figure out how much Ragnar Lodbrok needs to pay in total?

major **components** in the different ERD diagrams

	Conceptual	Logical	Physical
Relationship Labels	Y		
Entity Names	Y	Y	
High-level Relationships	Y	Y	Y
Cardinality of Relationships	Y	Y	Y
Optionality of Relationships	Y	Y	Y
Attributes		Y	
Attribute Preliminary Data		Y	
Primary keys		Y	Y
Foreign keys		Y	Y
Table Names			Y
Column Names			Y
Column Data Types (DB-specific)			Y
Column Constraints (DB-specific)			Y