PostgreSQL Physical Model

Schema for:

Model name: CLImodel
Author:
Version:
File name: CLImodel.json
File path: /Users/rvanbruggen/Library/CloudStorage/OneDrive-Hackolade/Demo/HackoladeGithubRepo/CLI/CLImodel.json
Printed On: Wed May 31 2023 15:31:02 GMT+0200 (Central European Summer Time)
Created with: Hackolade - Polyglot data modeling for NoSQL databases, storage formats, REST APIs, and JSON in RDBMS

1. Model

2. Schemas

3. Relationships

1. MODEL

1.1 Model CLImodel

1.1.1 CLImodel Entity Relationship Diagram
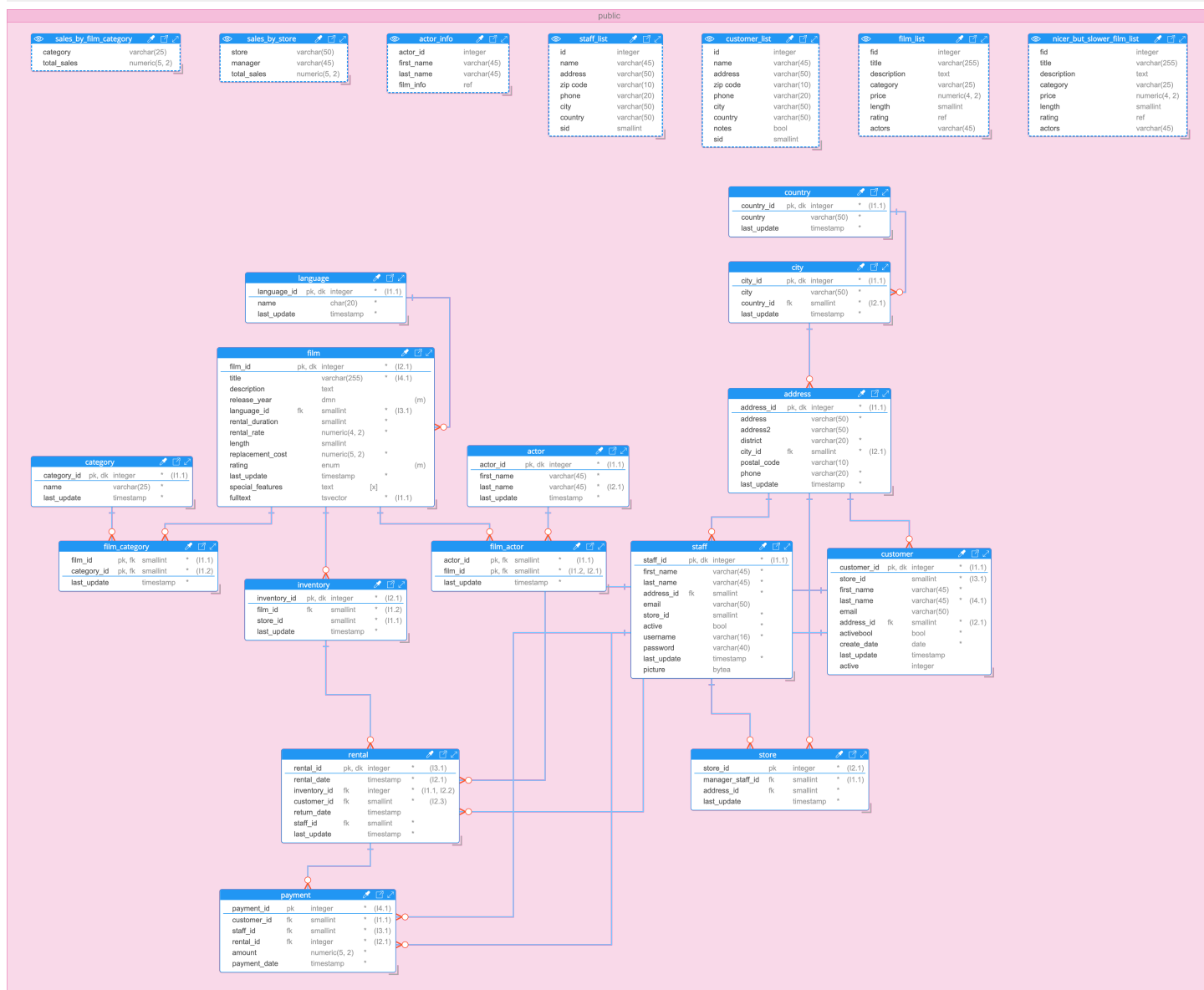
1.1.2 CLImodel Properties

1.1.2.1 Details tab

PROPERTY	VALUE
Model name	CLImodel
Technical name	
Description	
Author	
Version	
Synchronization Id	
DB vendor	PostgreSQL
DB version	v5.0
Database name	dvdrental
Tablespace	pg_default
Encoding	UTF8
Template	
Locale	
Collation	en_US.UTF-8
Character type	en_US.UTF-8
Lineage capture	
Polyglot models	
Comments	

1.1.3 CLImodel User-Defined Types

1.1.3.1 Column mpaa_rating

1.1.3.1.1 mpaa_rating Tree Diagram

1.1.3.1.2 mpaa_rating properties

PROPERTY	VALUE
Business Name	mpaa_rating
Technical name	
Id	
Type	enum
Comments	
Not null	
Pattern	
Default	
Enum	G,PG,PG-13,R,NC-17
Faker function	
Sample	
Remarks	

1.1.3.2 Column year

1.1.3.2.1 year Tree Diagram

1.1.3.2.2 year properties

PROPERTY	VALUE
Business Name	year
Technical name	
Id	
Type	domain
Underlying type	integer
Collation	
Not null	false
Default	
Check constraints	
[1] Constraint name	year_check
Check expression	((VALUE >= 1901) AND (VALUE <= 2155))
Comments	
Enum	
Faker function	
Sample	
Remarks	

2. SCHEMAS

2.1 Schema public
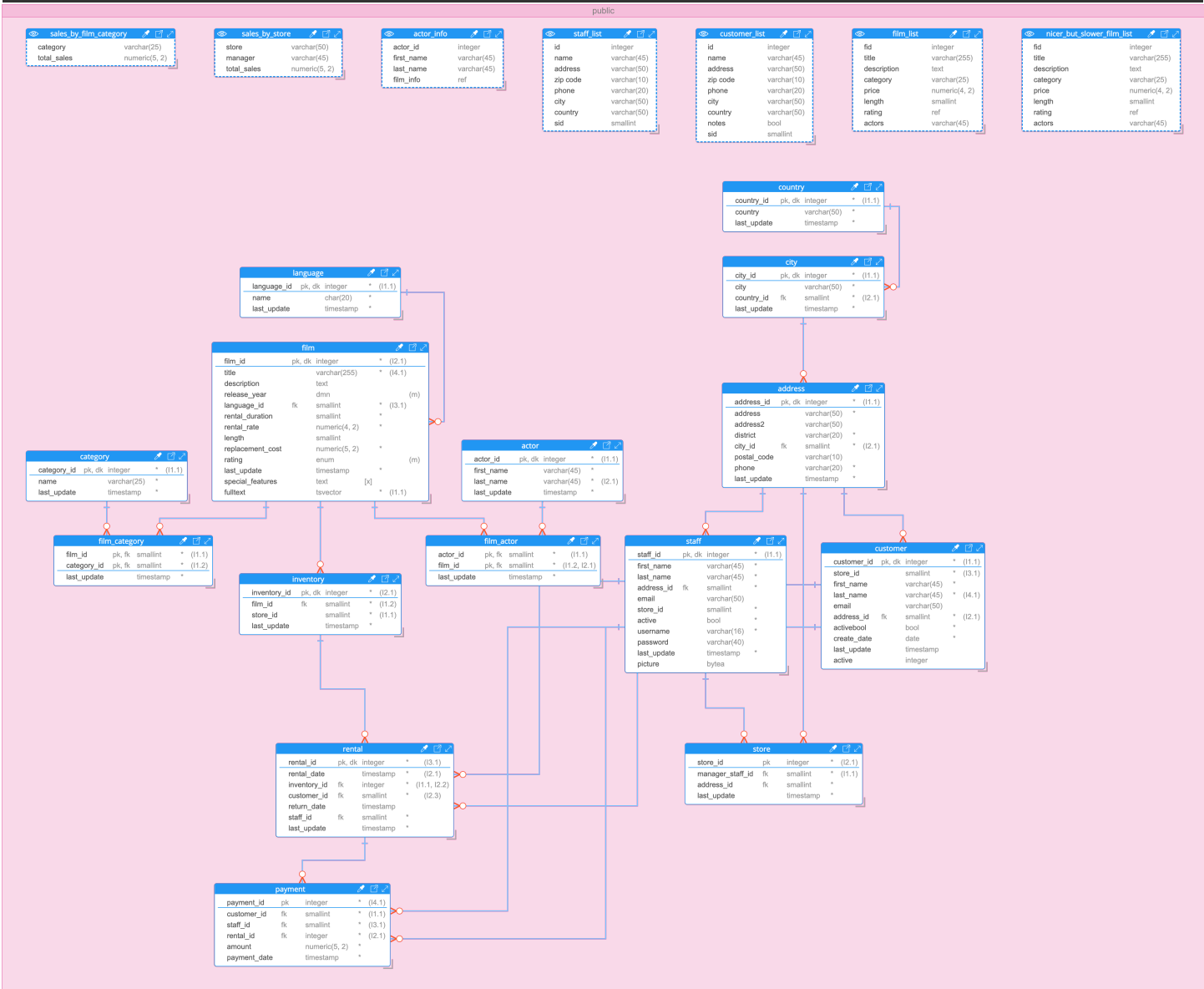
2.1.1 public Properties

PROPERTY	VALUE
Schema name	public
Technical name	
Activated	true
Comments	
If not exist	true
Remarks	

2.1.2 public Tables

2.1.2.1 Table actor

2.1.2.1.1 actor Properties

PROPERTY	VALUE
Table	actor
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.1.2 actor Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
actor_id	integer	true	pk, dk		
first_name	varchar(45)	true			
last_name	varchar(45)	true			
last_update	timestamp	true			

2.1.2.1.2.1 Column actor_id

2.1.2.1.2.1.1 actor_id Tree Diagram

2.1.2.1.2.1.2 actor_id properties

PROPERTY	VALUE
Business Name	actor_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('actor_actor_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	actor_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.1.2.2 Column first_name

2.1.2.1.2.2.1 first_name Tree Diagram

{ABC}   first_name

char(45)

2.1.2.1.2.2.2 first_name properties

PROPERTY	VALUE
Business Name	first_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.1.2.3 Column last_name

2.1.2.1.2.3.1 last_name Tree Diagram

{ABC}	last_name
(I2.1)	char(45)

2.1.2.1.2.3.2 last_name properties

PROPERTY	VALUE
Business Name	last_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.1.2.4 Column last_update

2.1.2.1.2.4.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.1.2.4.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.1.3 actor Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.1.4 actor Indexes

2.1.2.1.4.1 Index

PROPERTY	
Name	actor_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	actor_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.1.4.2 Index

PROPERTY	
Name	idx_actor_last_name
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	last_name
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.1.5 actor Triggers

2.1.2.1.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.1.6 actor JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "actor",
    "properties": {
        "actor_id": {
            "type": "number",
            "default": "nextval('actor_actor_id_seq'::regclass)"
        },
        "first_name": {
            "type": "string"
        },
        "last_name": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "actor_id",
        "first_name",
        "last_name",
        "last_update"
    ]
}
```

2.1.2.1.7 actor JSON data

```
{
    "actor_id": "nextval('actor_actor_id_seq'::regclass)",
    "first_name": "Lorem",
    "last_name": "Lorem",
    "last_update": "now()"
}
```

2.1.2.1.8 actor Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.actor (
    actor_id integer DEFAULT nextval('actor_actor_id_seq'::regclass) NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT actor_pkey PRIMARY KEY (actor_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.actor
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS actor_pkey
 ON ONLY public.actor USING BTREE (actor_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_actor_last_name
 ON ONLY public.actor USING BTREE (last_name COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;
```

2.1.2.2 Table address

2.1.2.2.1 address Properties

PROPERTY	VALUE
Table	address
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.2.2 address Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
address_id	integer	true	pk, dk		
address	varchar(50)	true			
address2	varchar(50)	false			
district	varchar(20)	true			
city_id	smallint	true	fk		
postal_code	varchar(10)	false			
phone	varchar(20)	true			
last_update	timestamp	true			

2.1.2.2.2.1 Column address_id

2.1.2.2.2.1.1 address_id Tree Diagram

2.1.2.2.2.1.2 address_id properties

PROPERTY	VALUE
Business Name	address_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('address_address_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	address_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.2 Column address

2.1.2.2.2.2.1 address Tree Diagram

2.1.2.2.2.2.2 address properties

PROPERTY	VALUE
Business Name	address
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.3 Column address2

2.1.2.2.2.3.1 address2 Tree Diagram

2.1.2.2.2.3.2 address2 properties

PROPERTY	VALUE
Business Name	address2
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.4 Column district

2.1.2.2.2.4.1 district Tree Diagram

2.1.2.2.2.4.2 district properties

PROPERTY	VALUE
Business Name	district
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	20
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.5 Column city_id

2.1.2.2.2.5.1 city_id Tree Diagram

{123}	city_id
(l2.1)	numeric

2.1.2.2.2.5.2 city_id properties

PROPERTY	VALUE
Business Name	city_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	city
Foreign field	city_id
Relationship type	Foreign Key
Relationship name	fk_address_city
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.6 Column postal_code

2.1.2.2.2.6.1 postal_code Tree Diagram

{ABC}   **postal_code**

char(10)

2.1.2.2.2.6.2 postal_code properties

PROPERTY	VALUE
Business Name	postal_code
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	10
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.7 Column phone

2.1.2.2.2.7.1 phone Tree Diagram

{ABC}	phone
	char(20)

2.1.2.2.2.7.2 phone properties

PROPERTY	VALUE
Business Name	phone
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	20
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.2.8 Column last_update

2.1.2.2.2.8.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.2.2.8.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.2.3 address Composite keys

PROPERTY		VALUE
Primary key		
[1] Constraint name		
Key		
Include non-key columns		
With storage parameters		
Index tablespace		
Comment		
Unique key		
[1] Constraint name		
Key		
Include non-key columns		
With storage parameters		
Index tablespace		
Comment		
Nulls Distinct		

2.1.2.2.4 address Indexes

2.1.2.2.4.1 Index

PROPERTY	
Name	address_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	address_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.2.4.2 Index

PROPERTY	
Name	idx_fk_city_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	city_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.2.5 address Triggers

2.1.2.2.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.2.6 address JSON Schema

```json
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "address",
    "properties": {
        "address_id": {
            "type": "number",
            "default": "nextval('address_address_id_seq'::regclass)"
        },
        "address": {
            "type": "string"
        },
        "address2": {
            "type": "string"
        },
        "district": {
            "type": "string"
        },
        "city_id": {
            "type": "number"
        },
        "postal_code": {
            "type": "string"
        },
        "phone": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "address_id",
        "address",
        "district",
        "city_id",
        "phone",
        "last_update"
    ]
}
```

2.1.2.2.7 address JSON data

```json
{
    "address_id": "nextval('address_address_id_seq'::regclass)",
    "address": "Lorem",
    "address2": "Lorem",
    "district": "Lorem",
    "city_id": -34,
    "postal_code": "Lorem",
    "phone": "Lorem",
    "last_update": "now()"
}
```

2.1.2.2.8 address Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;


/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */


/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */


CREATE TABLE IF NOT EXISTS public.address (
    address_id integer DEFAULT nextval('address_address_id_seq'::regclass) NOT NULL,
    address varchar(50) NOT NULL,
    address2 varchar(50),
    district varchar(20) NOT NULL,
    city_id smallint NOT NULL,
    postal_code varchar(10),
    phone varchar(20) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT address_pkey PRIMARY KEY (address_id),
    CONSTRAINT fk_address_city FOREIGN KEY (city_id) REFERENCES public.city (city_id) MATCH SIMPLE ON DELETE NO
ACTION ON UPDATE NO ACTION
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.address
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS address_pkey
 ON ONLY public.address USING BTREE (address_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_city_id
 ON ONLY public.address USING BTREE (city_id pg_catalog.int2_ops ASC NULLS LAST) ;
```

2.1.2.3 Table category

2.1.2.3.1 category Properties

PROPERTY	VALUE
Table	category
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.3.2 category Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
category_id	integer	true	pk, dk		
name	varchar(25)	true			
last_update	timestamp	true			

2.1.2.3.2.1 Column category_id

2.1.2.3.2.1.1 category_id Tree Diagram

{123}	category_id
(l1.1)	numeric

2.1.2.3.2.1.2 category_id properties

PROPERTY	VALUE
Business Name	category_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('category_category_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	category_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.3.2.2 Column name

2.1.2.3.2.2.1 name Tree Diagram

2.1.2.3.2.2.2 name properties

PROPERTY	VALUE
Business Name	name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	25
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.3.2.3 Column last_update

2.1.2.3.2.3.1 last_update Tree Diagram

2.1.2.3.2.3.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.3.3 category Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.3.4 category Indexes

2.1.2.3.4.1 Index

PROPERTY	
Name	category_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	category_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.3.5 category Triggers

2.1.2.3.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.3.6 category JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "category",
    "properties": {
        "category_id": {
            "type": "number",
            "default": "nextval('category_category_id_seq'::regclass)"
        },
        "name": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "category_id",
        "name",
        "last_update"
    ]
}
```

2.1.2.3.7 category JSON data

```
{
    "category_id": "nextval('category_category_id_seq'::regclass)",
    "name": "Lorem",
    "last_update": "now()"
}
```

2.1.2.3.8 category Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.category (
    category_id integer DEFAULT nextval('category_category_id_seq'::regclass) NOT NULL,
    name varchar(25) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT category_pkey PRIMARY KEY (category_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.category
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS category_pkey
 ON ONLY public.category USING BTREE (category_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.4 Table city

2.1.2.4.1 city Properties

PROPERTY	VALUE
Table	city
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.4.2 city Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
city_id	integer	true	pk, dk		
city	varchar(50)	true			
country_id	smallint	true	fk		
last_update	timestamp	true			

2.1.2.4.2.1 Column city_id

2.1.2.4.2.1.1 city_id Tree Diagram

2.1.2.4.2.1.2 city_id properties

PROPERTY	VALUE
Business Name	city_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('city_city_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	city_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.4.2.2 Column city

2.1.2.4.2.2.1 city Tree Diagram

2.1.2.4.2.2.2 city properties

PROPERTY	VALUE
Business Name	city
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.4.2.3 Column country_id

2.1.2.4.2.3.1 country_id Tree Diagram

```
{123}        country_id
(I2.1)        numeric
```

2.1.2.4.2.3.2 country_id properties

PROPERTY	VALUE
Business Name	country_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	country
Foreign field	country_id
Relationship type	Foreign Key
Relationship name	fk_city
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.4.2.4 Column last_update

2.1.2.4.2.4.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.4.2.4.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.4.3 city Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.4.4 city Indexes

2.1.2.4.4.1 Index

PROPERTY	
Name	city_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	city_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.4.4.2 Index

PROPERTY	
Name	idx_fk_country_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	country_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.4.5 city Triggers

2.1.2.4.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.4.6 city JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "city",
    "properties": {
        "city_id": {
            "type": "number",
            "default": "nextval('city_city_id_seq'::regclass)"
        },
        "city": {
            "type": "string"
        },
        "country_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "city_id",
        "city",
        "country_id",
        "last_update"
    ]
}
```

2.1.2.4.7 city JSON data

```
{
    "city_id": "nextval('city_city_id_seq'::regclass)",
    "city": "Lorem",
    "country_id": 30,
    "last_update": "now()"
}
```

2.1.2.4.8 city Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.city (
    city_id integer DEFAULT nextval('city_city_id_seq'::regclass) NOT NULL,
    city varchar(50) NOT NULL,
    country_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT city_pkey PRIMARY KEY (city_id),
    CONSTRAINT fk_city FOREIGN KEY (country_id) REFERENCES public.country (country_id) MATCH SIMPLE ON DELETE
NO ACTION ON UPDATE NO ACTION
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.city
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS city_pkey
 ON ONLY public.city USING BTREE (city_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_country_id
 ON ONLY public.city USING BTREE (country_id pg_catalog.int2_ops ASC NULLS LAST) ;
```

2.1.2.5 Table country

2.1.2.5.1 country Properties

PROPERTY	VALUE
Table	country
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.5.2 country Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
country_id	integer	true	pk, dk		
country	varchar(50)	true			
last_update	timestamp	true			

2.1.2.5.2.1 Column country_id

2.1.2.5.2.1.1 country_id Tree Diagram

2.1.2.5.2.1.2 country_id properties

PROPERTY	VALUE
Business Name	country_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('country_country_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	country_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.5.2.2 Column country

2.1.2.5.2.2.1 country Tree Diagram

2.1.2.5.2.2.2 country properties

PROPERTY	VALUE
Business Name	country
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.5.2.3 Column last_update

2.1.2.5.2.3.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.5.2.3.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.5.3 country Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.5.4 country Indexes

2.1.2.5.4.1 Index

PROPERTY	
Name	country_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	country_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.5.5 country Triggers

2.1.2.5.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.5.6 country JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "country",
    "properties": {
        "country_id": {
            "type": "number",
            "default": "nextval('country_country_id_seq'::regclass)"
        },
        "country": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "country_id",
        "country",
        "last_update"
    ]
}
```

2.1.2.5.7 country JSON data

```
{
    "country_id": "nextval('country_country_id_seq'::regclass)",
    "country": "Lorem",
    "last_update": "now()"
}
```

2.1.2.5.8 country Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.country (
    country_id integer DEFAULT nextval('country_country_id_seq'::regclass) NOT NULL,
    country varchar(50) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT country_pkey PRIMARY KEY (country_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.country
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS country_pkey
 ON ONLY public.country USING BTREE (country_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.6 Table customer

2.1.2.6.1 customer Properties

PROPERTY	VALUE
Table	customer
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.6.2 customer Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
customer_id	integer	true	pk, dk		
store_id	smallint	true			
first_name	varchar(45)	true			
last_name	varchar(45)	true			
email	varchar(50)	false			
address_id	smallint	true	fk		
activebool	boolean	true			
create_date	date	true			
last_update	timestamp	false			
active	integer	false			

2.1.2.6.2.1 Column customer_id

2.1.2.6.2.1.1 customer_id Tree Diagram

```
{123}    customer_id
(I1.1)       numeric
```

2.1.2.6.2.1.2 customer_id properties

PROPERTY	VALUE
Business Name	customer_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('customer_customer_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	customer_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.2 Column store_id

2.1.2.6.2.2.1 store_id Tree Diagram

```
{123}        store_id
(I3.1)       numeric
```

2.1.2.6.2.2.2 store_id properties

PROPERTY	VALUE
Business Name	store_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.3 Column first_name

2.1.2.6.2.3.1 first_name Tree Diagram

```
{ABC}        first_name
             char(45)
```

2.1.2.6.2.3.2 first_name properties

PROPERTY	VALUE
Business Name	first_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.4 Column last_name

2.1.2.6.2.4.1 last_name Tree Diagram

{ABC}	last_name
(I4.1)	char(45)

2.1.2.6.2.4.2 last_name properties

PROPERTY	VALUE
Business Name	last_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.5 Column email

2.1.2.6.2.5.1 email Tree Diagram

2.1.2.6.2.5.2 email properties

PROPERTY	VALUE
Business Name	email
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.6 Column address_id

2.1.2.6.2.6.1 address_id Tree Diagram

2.1.2.6.2.6.2 address_id properties

PROPERTY	VALUE
Business Name	address_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	address
Foreign field	address_id
Relationship type	Foreign Key
Relationship name	customer_address_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.7 Column activebool

2.1.2.6.2.7.1 activebool Tree Diagram

2.1.2.6.2.7.2 activebool properties

PROPERTY	VALUE
Business Name	activebool
Technical name	
Activated	true
Id	
Type	boolean
Comments	
Array type	
Not null	true
Default	true
Sample	
Remarks	

2.1.2.6.2.8 Column create_date

2.1.2.6.2.8.1 create_date Tree Diagram

2.1.2.6.2.8.2 create_date properties

PROPERTY	VALUE
Business Name	create_date
Technical name	
Activated	true
Id	
Type	datetime
Subtype	date
Comments	
Array type	
Not null	true
Default	('now'::text)::date
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.9 Column last_update

2.1.2.6.2.9.1 last_update Tree Diagram

2.1.2.6.2.9.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	false
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.2.10 Column active

2.1.2.6.2.10.1 active Tree Diagram

2.1.2.6.2.10.2 active properties

PROPERTY	VALUE
Business Name	active
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.6.3 customer Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.6.4 customer Indexes

2.1.2.6.4.1 Index

PROPERTY	
Name	customer_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	customer_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.6.4.2 Index

PROPERTY	
Name	idx_fk_address_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	address_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.6.4.3 Index

PROPERTY	
Name	idx_fk_store_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	store_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.6.4.4 Index

PROPERTY	
Name	idx_last_name
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	last_name
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.6.5 customer Triggers

2.1.2.6.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.6.6 customer JSON Schema

```json
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "customer",
    "properties": {
        "customer_id": {
            "type": "number",
            "default": "nextval('customer_customer_id_seq'::regclass)"
        },
        "store_id": {
            "type": "number"
        },
        "first_name": {
            "type": "string"
        },
        "last_name": {
            "type": "string"
        },
        "email": {
            "type": "string"
        },
        "address_id": {
            "type": "number"
        },
        "activebool": {
            "type": "boolean",
            "default": true
        },
        "create_date": {
            "type": "string",
            "default": "('now'::text)::date"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        },
        "active": {
            "type": "number"
        }
    },
    "additionalProperties": true,
    "required": [
        "customer_id",
        "store_id",
        "first_name",
        "last_name",
        "address_id",
        "activebool",
        "create_date"
    ]
}
```

2.1.2.6.7 customer JSON data

```
{
    "customer_id": "nextval('customer_customer_id_seq'::regclass)",
    "store_id": 89,
    "first_name": "Lorem",
    "last_name": "Lorem",
    "email": "Lorem",
    "address_id": 15,
    "activebool": true,
    "create_date": "('now'::text)::date",
    "last_update": "now()",
    "active": -14
}
```

2.1.2.6.8 customer Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;


/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */


/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */


CREATE TABLE IF NOT EXISTS public.customer (
    customer_id integer DEFAULT nextval('customer_customer_id_seq'::regclass) NOT NULL,
    store_id smallint NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    email varchar(50),
    address_id smallint NOT NULL,
    activebool boolean DEFAULT true NOT NULL,
    create_date date DEFAULT E'(\'now\'::text)::date' NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()',
    active integer,
    CONSTRAINT customer_pkey PRIMARY KEY (customer_id),
    CONSTRAINT customer_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.customer
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS customer_pkey
 ON ONLY public.customer USING BTREE (customer_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_address_id
 ON ONLY public.customer USING BTREE (address_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_store_id
 ON ONLY public.customer USING BTREE (store_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_last_name
 ON ONLY public.customer USING BTREE (last_name COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;
```

2.1.2.7 Table film

2.1.2.7.1 film Properties

PROPERTY	VALUE
Table	film
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Enable TOAST autovacuum	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.7.2 film Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
film_id	integer	true	pk, dk		
title	varchar(255)	true			
description	text	false			
release_year	domain	false			
language_id	smallint	true	fk		
rental_duration	smallint	true			
rental_rate	numeric(4, 2)	true			
length	smallint	false			
replacement_cost	numeric(5, 2)	true			
rating	enum	false			
last_update	timestamp	true			
special_features	text	false			
fulltext	tsvector	true			

2.1.2.7.2.1 Column film_id

2.1.2.7.2.1.1 film_id Tree Diagram

2.1.2.7.2.1.2 film_id properties

PROPERTY	VALUE
Business Name	film_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('film_film_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	film_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.2 Column title

2.1.2.7.2.2.1 title Tree Diagram

2.1.2.7.2.2.2 title properties

PROPERTY	VALUE
Business Name	title
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	255
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.3 Column description

2.1.2.7.2.3.1 description Tree Diagram

2.1.2.7.2.3.2 description properties

PROPERTY	VALUE
Business Name	description
Technical name	
Activated	true
Id	
Type	char
Subtype	text
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.4 Column release_year

2.1.2.7.2.4.1 release_year Tree Diagram

2.1.2.7.2.4.2 release_year properties

PROPERTY	VALUE
Business Name	release_year
Technical name	
Activated	true
Reference type	model
Reference description	

2.1.2.7.2.5 Column language_id

2.1.2.7.2.5.1 language_id Tree Diagram

2.1.2.7.2.5.2 language_id properties

PROPERTY	VALUE
Business Name	language_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	language
Foreign field	language_id
Relationship type	Foreign Key
Relationship name	film_language_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.6 Column rental_duration

2.1.2.7.2.6.1 rental_duration Tree Diagram

2.1.2.7.2.6.2 rental_duration properties

PROPERTY	VALUE
Business Name	rental_duration
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	3
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.7 Column rental_rate

2.1.2.7.2.7.1 rental_rate Tree Diagram

```
{123}        rental_rate

      numeric(4, 2)
```

2.1.2.7.2.7.2 rental_rate properties

PROPERTY	VALUE
Business Name	rental_rate
Technical name	
Activated	true
Id	
Type	numeric
Subtype	numeric
Precision	4
Scale	2
Array type	
Not null	true
Default	4.99
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.8 Column length

2.1.2.7.2.8.1 length Tree Diagram

2.1.2.7.2.8.2 length properties

PROPERTY	VALUE
Business Name	length
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.9 Column replacement_cost

2.1.2.7.2.9.1 replacement_cost Tree Diagram

{123}   replacement_cost
numeric(5, 2)

2.1.2.7.2.9.2 replacement_cost properties

PROPERTY	VALUE
Business Name	replacement_cost
Technical name	
Activated	true
Id	
Type	numeric
Subtype	numeric
Precision	5
Scale	2
Array type	
Not null	true
Default	19.99
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.10 Column rating

2.1.2.7.2.10.1 rating Tree Diagram

{enum}   **rating**

ref: mpaa_rating (m)

                              enum
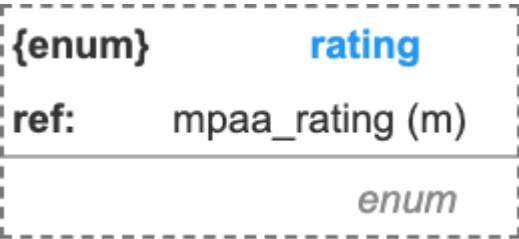
2.1.2.7.2.10.2 rating properties

PROPERTY	VALUE
Business Name	rating
Technical name	
Activated	true
Reference type	model
Reference description	

2.1.2.7.2.11 Column last_update

2.1.2.7.2.11.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.7.2.11.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.12 Column special_features

2.1.2.7.2.12.1 special_features Tree Diagram

{ABC}	special_features
	char

2.1.2.7.2.12.2 special_features properties

PROPERTY	VALUE
Business Name	special_features
Technical name	
Activated	true
Id	
Type	char
Subtype	text
Array type	
[1] Size limit	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.2.13 Column fulltext

2.1.2.7.2.13.1 fulltext Tree Diagram

```
{ABC}      fulltext
(I1.1)        char
```

2.1.2.7.2.13.2 fulltext properties

PROPERTY	VALUE
Business Name	fulltext
Technical name	
Activated	true
Id	
Type	char
Subtype	tsvector
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.7.3 film Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.7.4 film Indexes

2.1.2.7.4.1 Index

PROPERTY	
Name	film_fulltext_idx
Activated	true
Method	gist
Concurrent build	
If not exist	true
Only	true
Columns	
	fulltext
Include non-key columns	
With storage parameters	
[1] Fill factor	
Buffering	
Tablespace	
Where constraint	
Comment	

2.1.2.7.4.2 Index

PROPERTY	
Name	film_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	film_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.7.4.3 Index

PROPERTY	
Name	idx_fk_language_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	language_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.7.4.4 Index

PROPERTY	
Name	idx_title
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	title
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.7.5 film Triggers

2.1.2.7.5.1 Triggers film_fulltext_trigger

PROPERTY	FILM_FULLTEXT_TRIGGER	
Name	film_fulltext_trigger	
Description		
Or replace		
Constraint		
Trigger type	BEFORE	
Trigger events		
[1] Event	INSERT	
[2] Event	UPDATE	
Update columns		
Referencing		
Trigger for each row/statement	FOR EACH ROW	
Trigger WHEN condition		
Function	tsvector_update_trigger('fulltext', 'pg_catalog.english', 'title', 'description')	
Remarks		

2.1.2.7.5.2 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.7.6 film JSON Schema

```json
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "film",
    "properties": {
        "film_id": {
            "type": "number",
            "default": "nextval('film_film_id_seq'::regclass)"
        },
        "title": {
            "type": "string"
        },
        "description": {
            "type": "string"
        },
        "release_year": {
            "$ref": "#model/definitions/year"
        },
        "language_id": {
            "type": "number"
        },
        "rental_duration": {
            "type": "number",
            "default": "3"
        },
        "rental_rate": {
            "type": "number",
            "default": "4.99"
        },
        "length": {
            "type": "number"
        },
        "replacement_cost": {
            "type": "number",
            "default": "19.99"
        },
        "rating": {
            "$ref": "#model/definitions/mpaa_rating"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        },
        "special_features": {
            "type": "string"
        },
        "fulltext": {
            "type": "string"
        }
    },
    "additionalProperties": true,
    "required": [
        "film_id",
        "title",
        "language_id",
        "rental_duration",
        "rental_rate",
        "replacement_cost",
        "last_update",
        "fulltext"
```

```
    ]
}
```

2.1.2.7.7 film JSON data

```
{
    "film_id": "nextval('film_film_id_seq'::regclass)",
    "title": "Lorem",
    "description": "Lorem",
    "release_year": "Lorem",
    "language_id": -1,
    "rental_duration": "3",
    "rental_rate": "4.99",
    "length": 90,
    "replacement_cost": "19.99",
    "rating": "NC-17",
    "last_update": "now()",
    "special_features": "Lorem",
    "fulltext": "Lorem"
}
```

2.1.2.7.8 film Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');


CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

CREATE TABLE IF NOT EXISTS public.film (
    film_id integer DEFAULT nextval('film_film_id_seq'::regclass) NOT NULL,
    title varchar(255) NOT NULL,
    description text,
    release_year year,
    language_id smallint NOT NULL,
    rental_duration smallint DEFAULT 3 NOT NULL,
    rental_rate numeric(4,2) DEFAULT 4.99 NOT NULL,
    length smallint,
    replacement_cost numeric(5,2) DEFAULT 19.99 NOT NULL,
    rating mpaa_rating,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    special_features text[],
    fulltext tsvector NOT NULL,
    CONSTRAINT film_pkey PRIMARY KEY (film_id),
    CONSTRAINT film_language_id_fkey FOREIGN KEY (language_id) REFERENCES public.language (language_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER film_fulltext_trigger BEFORE INSERT OR UPDATE
    ON public.film
    FOR EACH ROW
    EXECUTE FUNCTION tsvector_update_trigger('fulltext', 'pg_catalog.english', 'title', 'description');

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE INDEX IF NOT EXISTS film_fulltext_idx
 ON ONLY public.film USING GIST (fulltext pg_catalog.tsvector_ops) ;

CREATE UNIQUE INDEX IF NOT EXISTS film_pkey
 ON ONLY public.film USING BTREE (film_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_language_id
 ON ONLY public.film USING BTREE (language_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_title
 ON ONLY public.film USING BTREE (title COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;
```

2.1.2.8 Table film_actor

2.1.2.8.1 film_actor Properties

PROPERTY	VALUE
Table	film_actor
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.8.2 film_actor Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
actor_id	smallint	true	pk, fk		
film_id	smallint	true	pk, fk		
last_update	timestamp	true			

2.1.2.8.2.1 Column actor_id

2.1.2.8.2.1.1 actor_id Tree Diagram

2.1.2.8.2.1.2 actor_id properties

PROPERTY	VALUE
Business Name	actor_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	true
Foreign table	actor
Foreign field	actor_id
Relationship type	Foreign Key
Relationship name	film_actor_actor_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.8.2.2 Column film_id

2.1.2.8.2.2.1 film_id Tree Diagram

2.1.2.8.2.2.2 film_id properties

PROPERTY	VALUE
Business Name	film_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	true
Foreign table	film
Foreign field	film_id
Relationship type	Foreign Key
Relationship name	film_actor_film_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.8.2.3 Column last_update

2.1.2.8.2.3.1 last_update Tree Diagram

{dt} **last_update**

datetime

2.1.2.8.2.3.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.8.3 film_actor Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	film_actor_pkey
Key	
	actor_id
	film_id
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.8.4 film_actor Indexes

2.1.2.8.4.1 Index

PROPERTY	
Name	film_actor_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	actor_id
	film_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.8.4.2 Index

PROPERTY	
Name	idx_fk_film_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	film_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.8.5 film_actor Triggers

2.1.2.8.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.8.6 film_actor JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "film_actor",
    "properties": {
        "actor_id": {
            "type": "number"
        },
        "film_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "actor_id",
        "film_id",
        "last_update"
    ]
}
```

2.1.2.8.7 film_actor JSON data

```
{
    "actor_id": -49,
    "film_id": -76,
    "last_update": "now()"
}
```

2.1.2.8.8 film_actor Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;


/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */


/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */


CREATE TABLE IF NOT EXISTS public.film_actor (
    actor_id smallint NOT NULL,
    film_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT film_actor_pkey PRIMARY KEY (actor_id, film_id),
    CONSTRAINT film_actor_actor_id_fkey FOREIGN KEY (actor_id) REFERENCES public.actor (actor_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT film_actor_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film_actor
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS film_actor_pkey
 ON ONLY public.film_actor USING BTREE (actor_id pg_catalog.int2_ops ASC NULLS LAST, film_id pg_catalog.int2_ops ASC
NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_film_id
 ON ONLY public.film_actor USING BTREE (film_id pg_catalog.int2_ops ASC NULLS LAST) ;
```

2.1.2.9 Table film_category

2.1.2.9.1 film_category Properties

PROPERTY	VALUE
Table	film_category
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.9.2 film_category Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
film_id	smallint	true	pk, fk		
category_id	smallint	true	pk, fk		
last_update	timestamp	true			

2.1.2.9.2.1 Column film_id

2.1.2.9.2.1.1 film_id Tree Diagram

2.1.2.9.2.1.2 film_id properties

PROPERTY	VALUE
Business Name	film_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	true
Foreign table	film
Foreign field	film_id
Relationship type	Foreign Key
Relationship name	film_category_film_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.9.2.2 Column category_id

2.1.2.9.2.2.1 category_id Tree Diagram

{123}   category_id
(I1.2)   numeric

2.1.2.9.2.2.2 category_id properties

PROPERTY	VALUE
Business Name	category_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	true
Foreign table	category
Foreign field	category_id
Relationship type	Foreign Key
Relationship name	film_category_category_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.9.2.3 Column last_update

2.1.2.9.2.3.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.9.2.3.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.9.3 film_category Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	film_category_pkey
Key	
	film_id
	category_id
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.9.4 film_category Indexes

2.1.2.9.4.1 Index

PROPERTY	
Name	film_category_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	film_id
	category_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.9.5 film_category Triggers

2.1.2.9.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.9.6 film_category JSON Schema

```json
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "film_category",
    "properties": {
        "film_id": {
            "type": "number"
        },
        "category_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "film_id",
        "category_id",
        "last_update"
    ]
}
```

2.1.2.9.7 film_category JSON data

```json
{
    "film_id": 2,
    "category_id": -87,
    "last_update": "now()"
}
```

2.1.2.9.8 film_category Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.film_category (
    film_id smallint NOT NULL,
    category_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT film_category_pkey PRIMARY KEY (film_id, category_id),
    CONSTRAINT film_category_category_id_fkey FOREIGN KEY (category_id) REFERENCES public.category (category_id)
MATCH SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT film_category_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film_category
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS film_category_pkey
 ON ONLY public.film_category USING BTREE (film_id pg_catalog.int2_ops ASC NULLS LAST, category_id pg_catalog.int2_ops
ASC NULLS LAST) ;
```

2.1.2.10 Table inventory

2.1.2.10.1 inventory Properties

PROPERTY	VALUE
Table	inventory
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.10.2 inventory Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
inventory_id	integer	true	pk, dk		
film_id	smallint	true	fk		
store_id	smallint	true			
last_update	timestamp	true			

2.1.2.10.2.1 Column inventory_id

2.1.2.10.2.1.1 inventory_id Tree Diagram

2.1.2.10.2.1.2 inventory_id properties

PROPERTY	VALUE
Business Name	inventory_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('inventory_inventory_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	inventory_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.10.2.2 Column film_id

2.1.2.10.2.2.1 film_id Tree Diagram

2.1.2.10.2.2.2 film_id properties

PROPERTY	VALUE
Business Name	film_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	film
Foreign field	film_id
Relationship type	Foreign Key
Relationship name	inventory_film_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.10.2.3 Column store_id

2.1.2.10.2.3.1 store_id Tree Diagram

{123}	store_id
(I1.1)	numeric

2.1.2.10.2.3.2 store_id properties

PROPERTY	VALUE
Business Name	store_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.10.2.4 Column last_update

2.1.2.10.2.4.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.10.2.4.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.10.3 inventory Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.10.4 inventory Indexes

2.1.2.10.4.1 Index

PROPERTY	
Name	idx_store_id_film_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	store_id
	film_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.10.4.2 Index

PROPERTY	
Name	inventory_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	inventory_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.10.5 inventory Triggers

2.1.2.10.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.10.6 inventory JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "inventory",
    "properties": {
        "inventory_id": {
            "type": "number",
            "default": "nextval('inventory_inventory_id_seq'::regclass)"
        },
        "film_id": {
            "type": "number"
        },
        "store_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "inventory_id",
        "film_id",
        "store_id",
        "last_update"
    ]
}
```

2.1.2.10.7 inventory JSON data

```
{
    "inventory_id": "nextval('inventory_inventory_id_seq'::regclass)",
    "film_id": 92,
    "store_id": -19,
    "last_update": "now()"
}
```

2.1.2.10.8 inventory Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.inventory (
    inventory_id integer DEFAULT nextval('inventory_inventory_id_seq'::regclass) NOT NULL,
    film_id smallint NOT NULL,
    store_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT inventory_pkey PRIMARY KEY (inventory_id),
    CONSTRAINT inventory_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.inventory
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE INDEX IF NOT EXISTS idx_store_id_film_id
 ON ONLY public.inventory USING BTREE (store_id pg_catalog.int2_ops ASC NULLS LAST, film_id pg_catalog.int2_ops ASC
NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS inventory_pkey
 ON ONLY public.inventory USING BTREE (inventory_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.11 Table language

2.1.2.11.1 language Properties

PROPERTY	VALUE
Table	language
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.11.2 language Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
language_id	integer	true	pk, dk		
name	char(20)	true			
last_update	timestamp	true			

2.1.2.11.2.1 Column language_id

2.1.2.11.2.1.1 language_id Tree Diagram

{123}	**language_id**
(I1.1)	*numeric*

2.1.2.11.2.1.2 language_id properties

PROPERTY	VALUE
Business Name	language_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('language_language_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	language_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.11.2.2 Column name

2.1.2.11.2.2.1 name Tree Diagram

2.1.2.11.2.2.2 name properties

PROPERTY	VALUE
Business Name	name
Technical name	
Activated	true
Id	
Type	char
Subtype	char
Length	20
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.11.2.3 Column last_update

2.1.2.11.2.3.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.11.2.3.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.11.3 language Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.11.4 language Indexes

2.1.2.11.4.1 Index

PROPERTY	
Name	language_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	language_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.11.5 language Triggers

2.1.2.11.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.11.6 language JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "language",
    "properties": {
        "language_id": {
            "type": "number",
            "default": "nextval('language_language_id_seq'::regclass)"
        },
        "name": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "language_id",
        "name",
        "last_update"
    ]
}
```

2.1.2.11.7 language JSON data

```
{
    "language_id": "nextval('language_language_id_seq'::regclass)",
    "name": "Lorem",
    "last_update": "now()"
}
```

2.1.2.11.8 language Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.language (
    language_id integer DEFAULT nextval('language_language_id_seq'::regclass) NOT NULL,
    name char(20) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT language_pkey PRIMARY KEY (language_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.language
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS language_pkey
 ON ONLY public.language USING BTREE (language_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.12 Table payment

2.1.2.12.1 payment Properties

PROPERTY	VALUE
Table	payment
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.12.2 payment Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
payment_id	integer	true	pk		
customer_id	smallint	true	fk		
staff_id	smallint	true	fk		
rental_id	integer	true	fk		
amount	numeric(5, 2)	true			
payment_date	timestamp	true			

2.1.2.12.2.1 Column payment_id

2.1.2.12.2.1.1 payment_id Tree Diagram

2.1.2.12.2.1.2 payment_id properties

PROPERTY	VALUE
Business Name	payment_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('payment_payment_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	payment_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.2.2 Column customer_id

2.1.2.12.2.2.1 customer_id Tree Diagram

2.1.2.12.2.2.2 customer_id properties

PROPERTY	VALUE
Business Name	customer_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	customer
Foreign field	customer_id
Relationship type	Foreign Key
Relationship name	payment_customer_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.2.3 Column staff_id

2.1.2.12.2.3.1 staff_id Tree Diagram

2.1.2.12.2.3.2 staff_id properties

PROPERTY	VALUE
Business Name	staff_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	staff
Foreign field	staff_id
Relationship type	Foreign Key
Relationship name	payment_staff_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.2.4 Column rental_id

2.1.2.12.2.4.1 rental_id Tree Diagram

2.1.2.12.2.4.2 rental_id properties

PROPERTY	VALUE
Business Name	rental_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	rental
Foreign field	rental_id
Relationship type	Foreign Key
Relationship name	payment_rental_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.2.5 Column amount

2.1.2.12.2.5.1 amount Tree Diagram

{123}   amount

numeric(5, 2)

2.1.2.12.2.5.2 amount properties

PROPERTY	VALUE
Business Name	amount
Technical name	
Activated	true
Id	
Type	numeric
Subtype	numeric
Precision	5
Scale	2
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.2.6 Column payment_date

2.1.2.12.2.6.1 payment_date Tree Diagram

{dt}	payment_date
	datetime

2.1.2.12.2.6.2 payment_date properties

PROPERTY	VALUE
Business Name	payment_date
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.12.3 payment Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.12.4 payment Indexes

2.1.2.12.4.1 Index

PROPERTY	
Name	idx_fk_customer_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	customer_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.12.4.2 Index

PROPERTY	
Name	idx_fk_rental_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	rental_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.12.4.3 Index

PROPERTY	
Name	idx_fk_staff_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	staff_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.12.4.4 Index

PROPERTY	
Name	payment_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	payment_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.12.5 payment JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "payment",
    "properties": {
        "payment_id": {
            "type": "number",
            "default": "nextval('payment_payment_id_seq'::regclass)"
        },
        "customer_id": {
            "type": "number"
        },
        "staff_id": {
            "type": "number"
        },
        "rental_id": {
            "type": "number"
        },
        "amount": {
            "type": "number"
        },
        "payment_date": {
            "type": "string"
        }
    },
    "additionalProperties": true,
    "required": [
        "payment_id",
        "customer_id",
        "staff_id",
        "rental_id",
        "amount",
        "payment_date"
    ]
}
```

2.1.2.12.6 payment JSON data

```
{
    "payment_id": "nextval('payment_payment_id_seq'::regclass)",
    "customer_id": -54,
    "staff_id": -12,
    "rental_id": 91,
    "amount": -97,
    "payment_date": "2011-02-03 04:05:00+0000"
}
```

2.1.2.12.7 payment Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */


/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */


CREATE TABLE IF NOT EXISTS public.payment (
    payment_id integer DEFAULT nextval('payment_payment_id_seq'::regclass) NOT NULL,
    customer_id smallint NOT NULL,
    staff_id smallint NOT NULL,
    rental_id integer NOT NULL,
    amount numeric(5,2) NOT NULL,
    payment_date timestamp WITHOUT TIME ZONE NOT NULL,
    CONSTRAINT payment_pkey PRIMARY KEY (payment_id),
    CONSTRAINT payment_customer_id_fkey FOREIGN KEY (customer_id) REFERENCES public.customer (customer_id)
MATCH SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT payment_rental_id_fkey FOREIGN KEY (rental_id) REFERENCES public.rental (rental_id) MATCH SIMPLE
ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT payment_staff_id_fkey FOREIGN KEY (staff_id) REFERENCES public.staff (staff_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE INDEX IF NOT EXISTS idx_fk_customer_id
 ON ONLY public.payment USING BTREE (customer_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_rental_id
 ON ONLY public.payment USING BTREE (rental_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_staff_id
 ON ONLY public.payment USING BTREE (staff_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS payment_pkey
 ON ONLY public.payment USING BTREE (payment_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.13 Table rental

2.1.2.13.1 rental Properties

PROPERTY	VALUE
Table	rental
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.13.2 rental Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
rental_id	integer	true	pk, dk		
rental_date	timestamp	true			
inventory_id	integer	true	fk		
customer_id	smallint	true	fk		
return_date	timestamp	false			
staff_id	smallint	true	fk		
last_update	timestamp	true			

2.1.2.13.2.1 Column rental_id

2.1.2.13.2.1.1 rental_id Tree Diagram

2.1.2.13.2.1.2 rental_id properties

PROPERTY	VALUE
Business Name	rental_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('rental_rental_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	rental_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.2 Column rental_date

2.1.2.13.2.2.1 rental_date Tree Diagram

2.1.2.13.2.2.2 rental_date properties

PROPERTY	VALUE
Business Name	rental_date
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.3 Column inventory_id

2.1.2.13.2.3.1 inventory_id Tree Diagram

2.1.2.13.2.3.2 inventory_id properties

PROPERTY	VALUE
Business Name	inventory_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	inventory
Foreign field	inventory_id
Relationship type	Foreign Key
Relationship name	rental_inventory_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.4 Column customer_id

2.1.2.13.2.4.1 customer_id Tree Diagram

{123}	customer_id
(I2.3)	numeric

2.1.2.13.2.4.2 customer_id properties

PROPERTY	VALUE
Business Name	customer_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	customer
Foreign field	customer_id
Relationship type	Foreign Key
Relationship name	rental_customer_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.5 Column return_date

2.1.2.13.2.5.1 return_date Tree Diagram

2.1.2.13.2.5.2 return_date properties

PROPERTY	VALUE
Business Name	return_date
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	false
Default	
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.6 Column staff_id

2.1.2.13.2.6.1 staff_id Tree Diagram

2.1.2.13.2.6.2 staff_id properties

PROPERTY	VALUE
Business Name	staff_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	staff
Foreign field	staff_id
Relationship type	Foreign Key
Relationship name	rental_staff_id_key
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.2.7 Column last_update

2.1.2.13.2.7.1 last_update Tree Diagram

2.1.2.13.2.7.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.13.3 rental Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.13.4 rental Indexes

2.1.2.13.4.1 Index

PROPERTY	
Name	idx_fk_inventory_id
Activated	true
Method	btree
Unique	
Concurrent build	
If not exist	true
Only	true
Columns	
	inventory_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.13.4.2 Index

PROPERTY	
Name	idx_unq_rental_rental_date_inventory_id_customer_id
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	rental_date
	inventory_id
	customer_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.13.4.3 Index

PROPERTY	
Name	rental_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	rental_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.13.5 rental Triggers

2.1.2.13.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.13.6 rental JSON Schema

```json
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "rental",
    "properties": {
        "rental_id": {
            "type": "number",
            "default": "nextval('rental_rental_id_seq'::regclass)"
        },
        "rental_date": {
            "type": "string"
        },
        "inventory_id": {
            "type": "number"
        },
        "customer_id": {
            "type": "number"
        },
        "return_date": {
            "type": "string"
        },
        "staff_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "rental_id",
        "rental_date",
        "inventory_id",
        "customer_id",
        "staff_id",
        "last_update"
    ]
}
```

2.1.2.13.7 rental JSON data

```json
{
    "rental_id": "nextval('rental_rental_id_seq'::regclass)",
    "rental_date": "2011-02-03 04:05:00+0000",
    "inventory_id": 51,
    "customer_id": -72,
    "return_date": "2011-02-03 04:05:00+0000",
    "staff_id": -64,
    "last_update": "now()"
}
```

2.1.2.13.8 rental Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.rental (
    rental_id integer DEFAULT nextval('rental_rental_id_seq'::regclass) NOT NULL,
    rental_date timestamp WITHOUT TIME ZONE NOT NULL,
    inventory_id integer NOT NULL,
    customer_id smallint NOT NULL,
    return_date timestamp WITHOUT TIME ZONE,
    staff_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT rental_pkey PRIMARY KEY (rental_id),
    CONSTRAINT rental_customer_id_fkey FOREIGN KEY (customer_id) REFERENCES public.customer (customer_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT rental_inventory_id_fkey FOREIGN KEY (inventory_id) REFERENCES public.inventory (inventory_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT rental_staff_id_key FOREIGN KEY (staff_id) REFERENCES public.staff (staff_id) MATCH SIMPLE ON
DELETE NO ACTION ON UPDATE NO ACTION
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.rental
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE INDEX IF NOT EXISTS idx_fk_inventory_id
 ON ONLY public.rental USING BTREE (inventory_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS idx_unq_rental_rental_date_inventory_id_customer_id
 ON ONLY public.rental USING BTREE (rental_date pg_catalog.timestamp_ops ASC NULLS LAST, inventory_id
pg_catalog.int4_ops ASC NULLS LAST, customer_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS rental_pkey
 ON ONLY public.rental USING BTREE (rental_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.14 Table staff

2.1.2.14.1 staff Properties

PROPERTY	VALUE
Table	staff
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Enable TOAST autovacuum	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.14.2 staff Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
staff_id	integer	true	pk, dk		
first_name	varchar(45)	true			
last_name	varchar(45)	true			
address_id	smallint	true	fk		
email	varchar(50)	false			
store_id	smallint	true			
active	boolean	true			
username	varchar(16)	true			
password	varchar(40)	false			
last_update	timestamp	true			
picture	bytea	false			

2.1.2.14.2.1 Column staff_id

2.1.2.14.2.1.1 staff_id Tree Diagram

2.1.2.14.2.1.2 staff_id properties

PROPERTY	VALUE
Business Name	staff_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('staff_staff_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	staff_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.2 Column first_name

2.1.2.14.2.2.1 first_name Tree Diagram

{ABC}	first_name
	char(45)

2.1.2.14.2.2.2 first_name properties

PROPERTY	VALUE
Business Name	first_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.3 Column last_name

2.1.2.14.2.3.1 last_name Tree Diagram

{ABC}	last_name
	char(45)

2.1.2.14.2.3.2 last_name properties

PROPERTY	VALUE
Business Name	last_name
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	45
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.4 Column address_id

2.1.2.14.2.4.1 address_id Tree Diagram

```
{123}        address_id
                 numeric
```

2.1.2.14.2.4.2 address_id properties

PROPERTY	VALUE
Business Name	address_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	address
Foreign field	address_id
Relationship type	Foreign Key
Relationship name	staff_address_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.5 Column email

2.1.2.14.2.5.1 email Tree Diagram

2.1.2.14.2.5.2 email properties

PROPERTY	VALUE
Business Name	email
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	50
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.6 Column store_id

2.1.2.14.2.6.1 store_id Tree Diagram

2.1.2.14.2.6.2 store_id properties

PROPERTY	VALUE
Business Name	store_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.7 Column active

2.1.2.14.2.7.1 active Tree Diagram
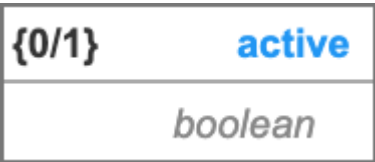
2.1.2.14.2.7.2 active properties

PROPERTY	VALUE
Business Name	active
Technical name	
Activated	true
Id	
Type	boolean
Comments	
Array type	
Not null	true
Default	true
Sample	
Remarks	

2.1.2.14.2.8 Column username

2.1.2.14.2.8.1 username Tree Diagram

2.1.2.14.2.8.2 username properties

PROPERTY	VALUE
Business Name	username
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	16
Array type	
Collation rule	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.9 Column password

2.1.2.14.2.9.1 password Tree Diagram

2.1.2.14.2.9.2 password properties

PROPERTY	VALUE
Business Name	password
Technical name	
Activated	true
Id	
Type	char
Subtype	varchar
Length	40
Array type	
Collation rule	
Not null	false
Default	
Comments	
Primary key	false
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Min length	
Max length	
Pattern	
Format	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.10 Column last_update

2.1.2.14.2.10.1 last_update Tree Diagram

{dt}	last_update
	datetime

2.1.2.14.2.10.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.14.2.11 Column picture

2.1.2.14.2.11.1 picture Tree Diagram

{BIN} picture

binary

2.1.2.14.2.11.2 picture properties

PROPERTY	VALUE
Business Name	picture
Technical name	
Activated	true
Id	
Type	binary
Subtype	bytea
Comments	
Array type	
Not null	false
Primary key	false
Unique	false
Remarks	

2.1.2.14.3 staff Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.14.4 staff Indexes

2.1.2.14.4.1 Index

PROPERTY	
Name	staff_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	staff_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.14.5 staff Triggers

2.1.2.14.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.14.6 staff JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "staff",
    "properties": {
        "staff_id": {
            "type": "number",
            "default": "nextval('staff_staff_id_seq'::regclass)"
        },
        "first_name": {
            "type": "string"
        },
        "last_name": {
            "type": "string"
        },
        "address_id": {
            "type": "number"
        },
        "email": {
            "type": "string"
        },
        "store_id": {
            "type": "number"
        },
        "active": {
            "type": "boolean",
            "default": true
        },
        "username": {
            "type": "string"
        },
        "password": {
            "type": "string"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        },
        "picture": {
            "type": "string"
        }
    },
    "additionalProperties": true,
    "required": [
        "staff_id",
        "first_name",
        "last_name",
        "address_id",
        "store_id",
        "active",
        "username",
        "last_update"
    ]
}
```

2.1.2.14.7 staff JSON data

```
{
    "staff_id": "nextval('staff_staff_id_seq'::regclass)",
    "first_name": "Lorem",
    "last_name": "Lorem",
    "address_id": 24,
    "email": "Lorem",
    "store_id": 16,
    "active": true,
    "username": "Lorem",
    "password": "Lorem",
    "last_update": "now()"
}
```

2.1.2.14.8 staff Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

*/

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

*/

CREATE TABLE IF NOT EXISTS public.staff (
    staff_id integer DEFAULT nextval('staff_staff_id_seq'::regclass) NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    address_id smallint NOT NULL,
    email varchar(50),
    store_id smallint NOT NULL,
    active boolean DEFAULT true NOT NULL,
    username varchar(16) NOT NULL,
    password varchar(40),
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    picture bytea,
    CONSTRAINT staff_pkey PRIMARY KEY (staff_id),
    CONSTRAINT staff_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.staff
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS staff_pkey
 ON ONLY public.staff USING BTREE (staff_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.2.15 Table store

2.1.2.15.1 store Properties

PROPERTY	VALUE
Table	store
Technical name	
Activated	true
Id	
Schema	public
Additional properties	true
$ref	
Comments	
Temporary	false
Unlogged	false
If not exists	true
Inherits parent tables	
[1] Table name	
Partition of	
Partition bounds	
Partitioning	
[1] Partition method	
Partition by	
Partition key	
Expression	
Using method	
Storage parameters	
[1] Fill factor	
Parallel workers	
Enable autovacuum	
Autovacuum params	
Enable TOAST autovacuum	
TOAST parameters	
User catalog table	
Tablespace	pg_default
As Select statement	
Remarks	

2.1.2.15.2 store Column

COLUMN	TYPE	REQ	KEY	DESCRIPTION	COMMENTS
store_id	integer	true	pk		
manager_staff_id	smallint	true	fk		
address_id	smallint	true	fk		
last_update	timestamp	true			

2.1.2.15.2.1 Column store_id

2.1.2.15.2.1.1 store_id Tree Diagram

2.1.2.15.2.1.2 store_id properties

PROPERTY	VALUE
Business Name	store_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	integer
Array type	
Not null	true
Default	nextval('store_store_id_seq'::regclass)
Comments	
Primary key	true
Primary key options	
[1] Constraint name	store_pkey
Include non-key columns	
With storage parameters	
Tablespace	
Comment	
Unique	false
Foreign table	
Foreign field	
Relationship type	
Relationship name	
Cardinality	
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.15.2.2 Column manager_staff_id

2.1.2.15.2.2.1 manager_staff_id Tree Diagram

{123} manager_staff_id

(I1.1) numeric

2.1.2.15.2.2.2 manager_staff_id properties

PROPERTY	VALUE
Business Name	manager_staff_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	staff
Foreign field	staff_id
Relationship type	Foreign Key
Relationship name	store_manager_staff_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.15.2.3 Column address_id

2.1.2.15.2.3.1 address_id Tree Diagram

2.1.2.15.2.3.2 address_id properties

PROPERTY	VALUE
Business Name	address_id
Technical name	
Activated	true
Id	
Type	numeric
Subtype	smallint
Array type	
Not null	true
Default	
Comments	
Primary key	false
Unique	false
Foreign table	address
Foreign field	address_id
Relationship type	Foreign Key
Relationship name	store_address_id_fkey
Cardinality	0..n
Unit	
Min value	
Excl min	false
Max value	
Excl max	false
Multiple of	
Divisible by	
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.15.2.4 Column last_update

2.1.2.15.2.4.1 last_update Tree Diagram

2.1.2.15.2.4.2 last_update properties

PROPERTY	VALUE
Business Name	last_update
Technical name	
Activated	true
Id	
Type	datetime
Subtype	timestamp
Precision	
Timezone	WITHOUT TIME ZONE
Comments	
Array type	
Not null	true
Default	now()
Primary key	false
Unique	false
Pattern	
Enum	
Faker function	
Sample	
Remarks	

2.1.2.15.3 store Composite keys

PROPERTY	VALUE
Primary key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Unique key	
[1] Constraint name	
Key	
Include non-key columns	
With storage parameters	
Index tablespace	
Comment	
Nulls Distinct	

2.1.2.15.4 store Indexes

2.1.2.15.4.1 Index

PROPERTY	
Name	idx_unq_manager_staff_id
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	manager_staff_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.15.4.2 Index

PROPERTY	
Name	store_pkey
Activated	true
Method	btree
Unique	true
Nulls Distinct	
Concurrent build	
If not exist	true
Only	true
Columns	
	store_id
Include non-key columns	
With storage parameters	
[1] Fill factor	
Deduplicate items	
Tablespace	
Where constraint	
Comment	

2.1.2.15.5 store Triggers

2.1.2.15.5.1 Triggers last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Description	
Or replace	
Constraint	
Trigger type	BEFORE
Trigger events	
[1] Event	UPDATE
Update columns	
Referencing	
Trigger for each row/statement	FOR EACH ROW
Trigger WHEN condition	
Function	last_updated()
Remarks	

2.1.2.15.6 store JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "store",
    "properties": {
        "store_id": {
            "type": "number",
            "default": "nextval('store_store_id_seq'::regclass)"
        },
        "manager_staff_id": {
            "type": "number"
        },
        "address_id": {
            "type": "number"
        },
        "last_update": {
            "type": "string",
            "default": "now()"
        }
    },
    "additionalProperties": true,
    "required": [
        "store_id",
        "manager_staff_id",
        "address_id",
        "last_update"
    ]
}
```

2.1.2.15.7 store JSON data

```
{
    "store_id": "nextval('store_store_id_seq'::regclass)",
    "manager_staff_id": -72,
    "address_id": -43,
    "last_update": "now()"
}
```

2.1.2.15.8 store Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

/*
CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');

 */

/*
CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

 */

CREATE TABLE IF NOT EXISTS public.store (
    store_id integer DEFAULT nextval('store_store_id_seq'::regclass) NOT NULL,
    manager_staff_id smallint NOT NULL,
    address_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT store_pkey PRIMARY KEY (store_id),
    CONSTRAINT store_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT store_manager_staff_id_fkey FOREIGN KEY (manager_staff_id) REFERENCES public.staff (staff_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.store
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS idx_unq_manager_staff_id
 ON ONLY public.store USING BTREE (manager_staff_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS store_pkey
 ON ONLY public.store USING BTREE (store_id pg_catalog.int4_ops ASC NULLS LAST) ;
```

2.1.3 public Functions

2.1.3.1 Functions _group_concat

PROPERTY	_GROUP_CONCAT
Name	_group_concat
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	
Argument type	text
Default expression	
[2] Arg mode	IN
Argument name	
Argument type	text
Default expression	
Returns set of	
Returns data type	text
Language	sql

Definition	SELECT CASE
	  WHEN $2 IS NULL THEN $1
	  WHEN $1 IS NULL THEN $2
	  ELSE $1 || ', ' || $2
	END

Volatility	IMMUTABLE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.2 Functions film_in_stock

PROPERTY	FILM_IN_STOCK
Name	film_in_stock
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	p_film_id
Argument type	integer
Default expression	
[2] Arg mode	IN
Argument name	p_store_id
Argument type	integer
Default expression	
[3] Arg mode	OUT
Argument name	p_film_count
Argument type	integer
Default expression	
Returns set of	true
Returns data type	int4
Language	sql
Definition	SELECT inventory_id FROM inventory WHERE film_id = $1 AND store_id = $2 AND inventory_in_stock(inventory_id);
Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Estimated rows	
Support function	
Config parameters	
Remarks	

2.1.3.3 Functions film_not_in_stock

PROPERTY	FILM_NOT_IN_STOCK
Name	film_not_in_stock
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	p_film_id
Argument type	integer
Default expression	
[2] Arg mode	IN
Argument name	p_store_id
Argument type	integer
Default expression	
[3] Arg mode	OUT
Argument name	p_film_count
Argument type	integer
Default expression	
Returns set of	true
Returns data type	int4
Language	sql
Definition	SELECT inventory_id FROM inventory WHERE film_id = $1 AND store_id = $2 AND NOT inventory_in_stock(inventory_id);
Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Estimated rows	
Support function	
Config parameters	
Remarks	

2.1.3.4 Functions get_customer_balance

PROPERTY	GET_CUSTOMER_BALANCE
Name	get_customer_balance
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	p_customer_id
Argument type	integer
Default expression	
[2] Arg mode	IN
Argument name	p_effective_date
Argument type	timestamp without time zone
Default expression	
Returns set of	
Returns data type	numeric
Language	plpgsql

Definition	`--#OK, WE NEED TO CALCULATE THE CURRENT BALANCE GIVEN A CUSTOMER_ID AND A DATE`
	`--#THAT WE WANT THE BALANCE TO BE EFFECTIVE FOR. THE BALANCE IS:`
	`--#   1) RENTAL FEES FOR ALL PREVIOUS RENTALS`
	`--#   2) ONE DOLLAR FOR EVERY DAY THE PREVIOUS RENTALS ARE OVERDUE`
	`--#   3) IF A FILM IS MORE THAN RENTAL_DURATION * 2 OVERDUE, CHARGE THE REPLACEMENT_COST`
	`--#   4) SUBTRACT ALL PAYMENTS MADE BEFORE THE DATE SPECIFIED`
	`DECLARE`
	`  v_rentfees DECIMAL(5,2); --#FEES PAID TO RENT THE VIDEOS INITIALLY`
	`  v_overfees INTEGER;      --#LATE FEES FOR PRIOR RENTALS`
	`  v_payments DECIMAL(5,2); --#SUM OF PAYMENTS MADE PREVIOUSLY`
	`BEGIN`
	`  SELECT COALESCE(SUM(film.rental_rate),0) INTO v_rentfees`
	`  FROM film, inventory, rental`
	`  WHERE film.film_id = inventory.film_id`
	`    AND inventory.inventory_id = rental.inventory_id`
	`    AND rental.rental_date <= p_effective_date`
	`    AND rental.customer_id = p_customer_id;`
	
	`  SELECT COALESCE(SUM(IF((rental.return_date - rental.rental_date) > (film.rental_duration * '1 day'::interval),`
	`    ((rental.return_date - rental.rental_date) - (film.rental_duration * '1 day'::interval)),0)),0)`
	`INTO v_overfees`
	`  FROM rental, inventory, film`
	`  WHERE film.film_id = inventory.film_id`
	`    AND inventory.inventory_id = rental.inventory_id`
	`    AND rental.rental_date <= p_effective_date`
	`    AND rental.customer_id = p_customer_id;`
	
	`  SELECT COALESCE(SUM(payment.amount),0) INTO v_payments`
	`  FROM payment`
	`  WHERE payment.payment_date <= p_effective_date`
	`  AND payment.customer_id = p_customer_id;`
	
	`  RETURN v_rentfees + v_overfees - v_payments;`
	`END`

Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT

PROPERTY	GET_CUSTOMER_BALANCE
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.5 Functions inventory_held_by_customer

PROPERTY	INVENTORY_HELD_BY_CUSTOMER
Name	inventory_held_by_customer
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	p_inventory_id
Argument type	integer
Default expression	
Returns set of	
Returns data type	int4
Language	plpgsql

Definition	DECLARE
	  v_customer_id INTEGER;
	BEGIN
	
	  SELECT customer_id INTO v_customer_id
	  FROM rental
	  WHERE return_date IS NULL
	  AND inventory_id = p_inventory_id;
	
	  RETURN v_customer_id;
	END

Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.6 Functions inventory_in_stock

PROPERTY	INVENTORY_IN_STOCK
Name	inventory_in_stock
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	p_inventory_id
Argument type	integer
Default expression	
Returns set of	
Returns data type	bool
Language	plpgsql

Definition	DECLARE
	  v_rentals INTEGER;
	  v_out INTEGER;
	BEGIN
	  -- AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE
	  -- FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED
	
	  SELECT count(*) INTO v_rentals
	  FROM rental
	  WHERE inventory_id = p_inventory_id;
	
	  IF v_rentals = 0 THEN
	    RETURN TRUE;
	  END IF;
	
	  SELECT COUNT(rental_id) INTO v_out
	  FROM inventory LEFT JOIN rental USING(inventory_id)
	  WHERE inventory.inventory_id = p_inventory_id
	  AND rental.return_date IS NULL;
	
	  IF v_out > 0 THEN
	    RETURN FALSE;
	  ELSE
	    RETURN TRUE;
	  END IF;
	END

Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.7 Functions last_day

PROPERTY	LAST_DAY
Name	last_day
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	
Argument type	timestamp without time zone
Default expression	
Returns set of	
Returns data type	date
Language	sql

Definition	SELECT CASE
	  WHEN EXTRACT(MONTH FROM $1) = 12 THEN
	    (((EXTRACT(YEAR FROM $1) + 1) operator(pg_catalog.||) '-01-01')::date - INTERVAL '1 day')::date
	  ELSE
	    ((EXTRACT(YEAR FROM $1) operator(pg_catalog.||) '-' operator(pg_catalog.||) (EXTRACT(MONTH FROM $1) + 1) operator(pg_catalog.||) '-01')::date - INTERVAL '1 day')::date
	  END

Volatility	IMMUTABLE
Leak proof	
When NULL args	STRICT
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.8 Functions last_updated

PROPERTY	LAST_UPDATED
Name	last_updated
Comments	
Or replace	
Arguments	
[1] Arg mode	
Argument name	
Argument type	
Default expression	
Returns set of	
Returns data type	trigger
Language	plpgsql

Definition	BEGIN
	   NEW.last_update = CURRENT_TIMESTAMP;
	   RETURN NEW;
	END

Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	INVOKER
Parallel	
Estimated cost	
Support function	
Config parameters	
Remarks	

2.1.3.9 Functions rewards_report

PROPERTY	REWARDS_REPORT
Name	rewards_report
Comments	
Or replace	
Arguments	
[1] Arg mode	IN
Argument name	min_monthly_purchases
Argument type	integer
Default expression	
[2] Arg mode	IN
Argument name	min_dollar_amount_purchased
Argument type	numeric
Default expression	
Returns set of	true
Returns data type	customer
Language	plpgsql

Definition	DECLARE
	    last_month_start DATE;
	    last_month_end DATE;
	rr RECORD;
	tmpSQL TEXT;
	BEGIN
	
	    /* Some sanity checks... */
	    IF min_monthly_purchases = 0 THEN
	        RAISE EXCEPTION 'Minimum monthly purchases parameter must be > 0';
	    END IF;
	    IF min_dollar_amount_purchased = 0.00 THEN
	        RAISE EXCEPTION 'Minimum monthly dollar amount purchased parameter must be > $0.00';
	    END IF;
	
	    last_month_start := CURRENT_DATE - '3 month'::interval;
	    last_month_start := to_date((extract(YEAR FROM last_month_start) || '-' || extract(MONTH FROM last_month_start) || '-01'),'YYYY-MM-DD');
	    last_month_end := LAST_DAY(last_month_start);
	
	    /*
	    Create a temporary storage area for Customer IDs.
	    */
	    CREATE TEMPORARY TABLE tmpCustomer (customer_id INTEGER NOT NULL PRIMARY KEY);
	
	    /*
	    Find all customers meeting the monthly purchase requirements
	    */
	
	    tmpSQL := 'INSERT INTO tmpCustomer (customer_id)
	        SELECT p.customer_id
	        FROM payment AS p
	        WHERE DATE(p.payment_date) BETWEEN '||quote_literal(last_month_start) ||' AND '|| quote_literal(last_month_end) || '
	        GROUP BY customer_id
	        HAVING SUM(p.amount) > '|| min_dollar_amount_purchased || '
	        AND COUNT(customer_id) > ' ||min_monthly_purchases ;

PROPERTY	REWARDS_REPORT
	EXECUTE tmpSQL; /* Output ALL customer information of matching rewardees. Customize output as needed. */ FOR rr IN EXECUTE 'SELECT c.* FROM tmpCustomer AS t INNER JOIN customer AS c ON t.customer_id = c.customer_id' LOOP RETURN NEXT rr; END LOOP; /* Clean up */ tmpSQL := 'DROP TABLE tmpCustomer'; EXECUTE tmpSQL; RETURN; END
Volatility	VOLATILE
Leak proof	
When NULL args	CALLED ON NULL INPUT
SQL Security	DEFINER
Parallel	
Estimated cost	
Estimated rows	
Support function	
Config parameters	
Remarks	

2.1.4 public Target Script

```
CREATE DATABASE dvdrental
    ENCODING UTF8
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TABLESPACE 'pg_default';

CREATE SCHEMA IF NOT EXISTS public;
SET search_path TO public;

CREATE FUNCTION public._group_concat
    (IN  text, IN  text)
    RETURNS text
    LANGUAGE sql
    IMMUTABLE
    NOT LEAKPROOF
    CALLED ON NULL INPUT
    SECURITY INVOKER
AS $BODY$

SELECT CASE
  WHEN $2 IS NULL THEN $1
  WHEN $1 IS NULL THEN $2
  ELSE $1 || ', ' || $2
END

$BODY$;

CREATE FUNCTION public.film_in_stock
    (IN p_film_id integer, IN p_store_id integer, OUT p_film_count integer)
    RETURNS SETOF int4
    LANGUAGE sql
    VOLATILE
    NOT LEAKPROOF
    CALLED ON NULL INPUT
    SECURITY INVOKER
AS $BODY$

    SELECT inventory_id
    FROM inventory
    WHERE film_id = $1
    AND store_id = $2
    AND inventory_in_stock(inventory_id);

$BODY$;

CREATE FUNCTION public.film_not_in_stock
    (IN p_film_id integer, IN p_store_id integer, OUT p_film_count integer)
    RETURNS SETOF int4
    LANGUAGE sql
    VOLATILE
    NOT LEAKPROOF
    CALLED ON NULL INPUT
    SECURITY INVOKER
AS $BODY$

    SELECT inventory_id
    FROM inventory
    WHERE film_id = $1
    AND store_id = $2
    AND NOT inventory_in_stock(inventory_id);
```

```
$BODY$;

CREATE FUNCTION public.get_customer_balance
     (IN p_customer_id integer, IN p_effective_date timestamp without time zone)
     RETURNS numeric
     LANGUAGE plpgsql
     VOLATILE
     NOT LEAKPROOF
     CALLED ON NULL INPUT
     SECURITY INVOKER
AS $BODY$

    --#OK, WE NEED TO CALCULATE THE CURRENT BALANCE GIVEN A CUSTOMER_ID AND A DATE
    --#THAT WE WANT THE BALANCE TO BE EFFECTIVE FOR. THE BALANCE IS:
    --#  1) RENTAL FEES FOR ALL PREVIOUS RENTALS
    --#  2) ONE DOLLAR FOR EVERY DAY THE PREVIOUS RENTALS ARE OVERDUE
    --#  3) IF A FILM IS MORE THAN RENTAL_DURATION * 2 OVERDUE, CHARGE THE REPLACEMENT_COST
    --#  4) SUBTRACT ALL PAYMENTS MADE BEFORE THE DATE SPECIFIED
DECLARE
  v_rentfees DECIMAL(5,2); --#FEES PAID TO RENT THE VIDEOS INITIALLY
  v_overfees INTEGER;      --#LATE FEES FOR PRIOR RENTALS
  v_payments DECIMAL(5,2); --#SUM OF PAYMENTS MADE PREVIOUSLY
BEGIN
  SELECT COALESCE(SUM(film.rental_rate),0) INTO v_rentfees
  FROM film, inventory, rental
  WHERE film.film_id = inventory.film_id
    AND inventory.inventory_id = rental.inventory_id
    AND rental.rental_date <= p_effective_date
    AND rental.customer_id = p_customer_id;

  SELECT COALESCE(SUM(IF((rental.return_date - rental.rental_date) > (film.rental_duration * '1 day'::interval),
    ((rental.return_date - rental.rental_date) - (film.rental_duration * '1 day'::interval)),0)),0) INTO v_overfees
  FROM rental, inventory, film
  WHERE film.film_id = inventory.film_id
    AND inventory.inventory_id = rental.inventory_id
    AND rental.rental_date <= p_effective_date
    AND rental.customer_id = p_customer_id;

  SELECT COALESCE(SUM(payment.amount),0) INTO v_payments
  FROM payment
  WHERE payment.payment_date <= p_effective_date
  AND payment.customer_id = p_customer_id;

  RETURN v_rentfees + v_overfees - v_payments;
END

$BODY$;

CREATE FUNCTION public.inventory_held_by_customer
     (IN p_inventory_id integer)
     RETURNS int4
     LANGUAGE plpgsql
     VOLATILE
     NOT LEAKPROOF
     CALLED ON NULL INPUT
     SECURITY INVOKER
AS $BODY$

DECLARE
  v_customer_id INTEGER;
```

```
 BEGIN

  SELECT customer_id INTO v_customer_id
  FROM rental
  WHERE return_date IS NULL
  AND inventory_id = p_inventory_id;

  RETURN v_customer_id;
 END
 $BODY$;

 CREATE FUNCTION public.inventory_in_stock
      (IN p_inventory_id integer)
      RETURNS bool
      LANGUAGE plpgsql
      VOLATILE
      NOT LEAKPROOF
      CALLED ON NULL INPUT
      SECURITY INVOKER
 AS $BODY$

 DECLARE
   v_rentals INTEGER;
   v_out     INTEGER;
 BEGIN
   -- AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE
   -- FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED

   SELECT count(*) INTO v_rentals
   FROM rental
   WHERE inventory_id = p_inventory_id;

   IF v_rentals = 0 THEN
     RETURN TRUE;
   END IF;

   SELECT COUNT(rental_id) INTO v_out
   FROM inventory LEFT JOIN rental USING(inventory_id)
   WHERE inventory.inventory_id = p_inventory_id
   AND rental.return_date IS NULL;

   IF v_out > 0 THEN
     RETURN FALSE;
   ELSE
     RETURN TRUE;
   END IF;
 END
 $BODY$;

 CREATE FUNCTION public.last_day
      (IN  timestamp without time zone)
      RETURNS date
      LANGUAGE sql
      IMMUTABLE
      NOT LEAKPROOF
      STRICT
      SECURITY INVOKER
 AS $BODY$

  SELECT CASE
    WHEN EXTRACT(MONTH FROM $1) = 12 THEN
```

```
    (((EXTRACT(YEAR FROM $1) + 1) operator(pg_catalog.||) '-01-01')::date - INTERVAL '1 day')::date
  ELSE
    ((EXTRACT(YEAR FROM $1) operator(pg_catalog.||) '-' operator(pg_catalog.||) (EXTRACT(MONTH FROM $1) + 1)
operator(pg_catalog.||) '-01')::date - INTERVAL '1 day')::date
  END

$BODY$;

CREATE FUNCTION public.last_updated
    ()
    RETURNS trigger
    LANGUAGE plpgsql
    VOLATILE
    NOT LEAKPROOF
    CALLED ON NULL INPUT
    SECURITY INVOKER
AS $BODY$

BEGIN
  NEW.last_update = CURRENT_TIMESTAMP;
  RETURN NEW;
END
$BODY$;

CREATE FUNCTION public.rewards_report
    (IN min_monthly_purchases integer, IN min_dollar_amount_purchased numeric)
    RETURNS SETOF customer
    LANGUAGE plpgsql
    VOLATILE
    NOT LEAKPROOF
    CALLED ON NULL INPUT
    SECURITY DEFINER
AS $BODY$

DECLARE
  last_month_start DATE;
  last_month_end DATE;
rr RECORD;
tmpSQL TEXT;
BEGIN

  /* Some sanity checks... */
  IF min_monthly_purchases = 0 THEN
    RAISE EXCEPTION 'Minimum monthly purchases parameter must be > 0';
  END IF;
  IF min_dollar_amount_purchased = 0.00 THEN
    RAISE EXCEPTION 'Minimum monthly dollar amount purchased parameter must be > $0.00';
  END IF;

  last_month_start := CURRENT_DATE - '3 month'::interval;
  last_month_start := to_date((extract(YEAR FROM last_month_start) || '-' || extract(MONTH FROM last_month_start) ||
'-01'),'YYYY-MM-DD');
  last_month_end := LAST_DAY(last_month_start);

  /*
  Create a temporary storage area for Customer IDs.
  */
  CREATE TEMPORARY TABLE tmpCustomer (customer_id INTEGER NOT NULL PRIMARY KEY);

  /*
  Find all customers meeting the monthly purchase requirements
```

```
    */

    tmpSQL := 'INSERT INTO tmpCustomer (customer_id)
        SELECT p.customer_id
        FROM payment AS p
        WHERE DATE(p.payment_date) BETWEEN '||quote_literal(last_month_start) ||' AND '|| quote_literal(last_month_end) || '
        GROUP BY customer_id
        HAVING SUM(p.amount) > '|| min_dollar_amount_purchased || '
        AND COUNT(customer_id) > ' ||min_monthly_purchases ;

    EXECUTE tmpSQL;

    /*
    Output ALL customer information of matching rewardees.
    Customize output as needed.
    */
    FOR rr IN EXECUTE 'SELECT c.* FROM tmpCustomer AS t INNER JOIN customer AS c ON t.customer_id = c.customer_id'
LOOP
        RETURN NEXT rr;
    END LOOP;

    /* Clean up */
    tmpSQL := 'DROP TABLE tmpCustomer';
    EXECUTE tmpSQL;

RETURN;
END

$BODY$;

CREATE TYPE public.mpaa_rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17');


CREATE DOMAIN public.year AS integer
    CONSTRAINT year_check CHECK (((VALUE >= 1901) AND (VALUE <= 2155)));

CREATE TABLE IF NOT EXISTS public.actor (
    actor_id integer DEFAULT nextval('actor_actor_id_seq'::regclass) NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT actor_pkey PRIMARY KEY (actor_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.actor
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS actor_pkey
 ON ONLY public.actor USING BTREE (actor_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_actor_last_name
 ON ONLY public.actor USING BTREE (last_name COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.country (
    country_id integer DEFAULT nextval('country_country_id_seq'::regclass) NOT NULL,
    country varchar(50) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT country_pkey PRIMARY KEY (country_id)
) TABLESPACE pg_default;
```

```
CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.country
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS country_pkey
 ON ONLY public.country USING BTREE (country_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.city (
    city_id integer DEFAULT nextval('city_city_id_seq'::regclass) NOT NULL,
    city varchar(50) NOT NULL,
    country_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT city_pkey PRIMARY KEY (city_id),
    CONSTRAINT fk_city FOREIGN KEY (country_id) REFERENCES public.country (country_id) MATCH SIMPLE ON DELETE
NO ACTION ON UPDATE NO ACTION
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.city
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS city_pkey
 ON ONLY public.city USING BTREE (city_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_country_id
 ON ONLY public.city USING BTREE (country_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.address (
    address_id integer DEFAULT nextval('address_address_id_seq'::regclass) NOT NULL,
    address varchar(50) NOT NULL,
    address2 varchar(50),
    district varchar(20) NOT NULL,
    city_id smallint NOT NULL,
    postal_code varchar(10),
    phone varchar(20) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT address_pkey PRIMARY KEY (address_id),
    CONSTRAINT fk_address_city FOREIGN KEY (city_id) REFERENCES public.city (city_id) MATCH SIMPLE ON DELETE NO
ACTION ON UPDATE NO ACTION
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.address
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS address_pkey
 ON ONLY public.address USING BTREE (address_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_city_id
 ON ONLY public.address USING BTREE (city_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.category (
    category_id integer DEFAULT nextval('category_category_id_seq'::regclass) NOT NULL,
    name varchar(25) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT category_pkey PRIMARY KEY (category_id)
) TABLESPACE pg_default;
```

```
CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.category
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS category_pkey
 ON ONLY public.category USING BTREE (category_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.customer (
    customer_id integer DEFAULT nextval('customer_customer_id_seq'::regclass) NOT NULL,
    store_id smallint NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    email varchar(50),
    address_id smallint NOT NULL,
    activebool boolean DEFAULT true NOT NULL,
    create_date date DEFAULT E'(\'now\'::text)::date' NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()',
    active integer,
    CONSTRAINT customer_pkey PRIMARY KEY (customer_id),
    CONSTRAINT customer_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.customer
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS customer_pkey
 ON ONLY public.customer USING BTREE (customer_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_address_id
 ON ONLY public.customer USING BTREE (address_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_store_id
 ON ONLY public.customer USING BTREE (store_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_last_name
 ON ONLY public.customer USING BTREE (last_name COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.language (
    language_id integer DEFAULT nextval('language_language_id_seq'::regclass) NOT NULL,
    name char(20) NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT language_pkey PRIMARY KEY (language_id)
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.language
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS language_pkey
 ON ONLY public.language USING BTREE (language_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.film (
    film_id integer DEFAULT nextval('film_film_id_seq'::regclass) NOT NULL,
    title varchar(255) NOT NULL,
    description text,
```

```
    release_year year,
    language_id smallint NOT NULL,
    rental_duration smallint DEFAULT 3 NOT NULL,
    rental_rate numeric(4,2) DEFAULT 4.99 NOT NULL,
    length smallint,
    replacement_cost numeric(5,2) DEFAULT 19.99 NOT NULL,
    rating mpaa_rating,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    special_features text[],
    fulltext tsvector NOT NULL,
    CONSTRAINT film_pkey PRIMARY KEY (film_id),
    CONSTRAINT film_language_id_fkey FOREIGN KEY (language_id) REFERENCES public.language (language_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER film_fulltext_trigger BEFORE INSERT OR UPDATE
    ON public.film
    FOR EACH ROW
    EXECUTE FUNCTION tsvector_update_trigger('fulltext', 'pg_catalog.english', 'title', 'description');

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE INDEX IF NOT EXISTS film_fulltext_idx
 ON ONLY public.film USING GIST (fulltext pg_catalog.tsvector_ops) ;

CREATE UNIQUE INDEX IF NOT EXISTS film_pkey
 ON ONLY public.film USING BTREE (film_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_language_id
 ON ONLY public.film USING BTREE (language_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_title
 ON ONLY public.film USING BTREE (title COLLATE pg_catalog."default" pg_catalog.text_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.film_actor (
    actor_id smallint NOT NULL,
    film_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT film_actor_pkey PRIMARY KEY (actor_id, film_id),
    CONSTRAINT film_actor_actor_id_fkey FOREIGN KEY (actor_id) REFERENCES public.actor (actor_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT film_actor_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film_actor
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS film_actor_pkey
 ON ONLY public.film_actor USING BTREE (actor_id pg_catalog.int2_ops ASC NULLS LAST, film_id pg_catalog.int2_ops ASC
NULLS LAST) ;

CREATE INDEX IF NOT EXISTS idx_fk_film_id
 ON ONLY public.film_actor USING BTREE (film_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.film_category (
```

```
    film_id smallint NOT NULL,
    category_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT film_category_pkey PRIMARY KEY (film_id, category_id),
    CONSTRAINT film_category_category_id_fkey FOREIGN KEY (category_id) REFERENCES public.category (category_id)
MATCH SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT film_category_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.film_category
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS film_category_pkey
 ON ONLY public.film_category USING BTREE (film_id pg_catalog.int2_ops ASC NULLS LAST, category_id pg_catalog.int2_ops
ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.inventory (
    inventory_id integer DEFAULT nextval('inventory_inventory_id_seq'::regclass) NOT NULL,
    film_id smallint NOT NULL,
    store_id smallint NOT NULL,
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    CONSTRAINT inventory_pkey PRIMARY KEY (inventory_id),
    CONSTRAINT inventory_film_id_fkey FOREIGN KEY (film_id) REFERENCES public.film (film_id) MATCH SIMPLE ON
DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.inventory
    FOR EACH ROW
    EXECUTE FUNCTION last_updated();

CREATE INDEX IF NOT EXISTS idx_store_id_film_id
 ON ONLY public.inventory USING BTREE (store_id pg_catalog.int2_ops ASC NULLS LAST, film_id pg_catalog.int2_ops ASC
NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS inventory_pkey
 ON ONLY public.inventory USING BTREE (inventory_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.staff (
    staff_id integer DEFAULT nextval('staff_staff_id_seq'::regclass) NOT NULL,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    address_id smallint NOT NULL,
    email varchar(50),
    store_id smallint NOT NULL,
    active boolean DEFAULT true NOT NULL,
    username varchar(16) NOT NULL,
    password varchar(40),
    last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
    picture bytea,
    CONSTRAINT staff_pkey PRIMARY KEY (staff_id),
    CONSTRAINT staff_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
    ON public.staff
    FOR EACH ROW
```

```
        EXECUTE FUNCTION last_updated();

 CREATE UNIQUE INDEX IF NOT EXISTS staff_pkey
  ON ONLY public.staff USING BTREE (staff_id pg_catalog.int4_ops ASC NULLS LAST) ;

 CREATE TABLE IF NOT EXISTS public.rental (
        rental_id integer DEFAULT nextval('rental_rental_id_seq'::regclass) NOT NULL,
        rental_date timestamp WITHOUT TIME ZONE NOT NULL,
        inventory_id integer NOT NULL,
        customer_id smallint NOT NULL,
        return_date timestamp WITHOUT TIME ZONE,
        staff_id smallint NOT NULL,
        last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
        CONSTRAINT rental_pkey PRIMARY KEY (rental_id),
        CONSTRAINT rental_customer_id_fkey FOREIGN KEY (customer_id) REFERENCES public.customer (customer_id) MATCH
 SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT rental_inventory_id_fkey FOREIGN KEY (inventory_id) REFERENCES public.inventory (inventory_id) MATCH
 SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT rental_staff_id_key FOREIGN KEY (staff_id) REFERENCES public.staff (staff_id) MATCH SIMPLE ON
 DELETE NO ACTION ON UPDATE NO ACTION
 ) TABLESPACE pg_default;

 CREATE TRIGGER last_updated BEFORE UPDATE
        ON public.rental
        FOR EACH ROW
        EXECUTE FUNCTION last_updated();

 CREATE INDEX IF NOT EXISTS idx_fk_inventory_id
  ON ONLY public.rental USING BTREE (inventory_id pg_catalog.int4_ops ASC NULLS LAST) ;

 CREATE UNIQUE INDEX IF NOT EXISTS idx_unq_rental_rental_date_inventory_id_customer_id
  ON ONLY public.rental USING BTREE (rental_date pg_catalog.timestamp_ops ASC NULLS LAST, inventory_id
 pg_catalog.int4_ops ASC NULLS LAST, customer_id pg_catalog.int2_ops ASC NULLS LAST) ;

 CREATE UNIQUE INDEX IF NOT EXISTS rental_pkey
  ON ONLY public.rental USING BTREE (rental_id pg_catalog.int4_ops ASC NULLS LAST) ;

 CREATE TABLE IF NOT EXISTS public.payment (
        payment_id integer DEFAULT nextval('payment_payment_id_seq'::regclass) NOT NULL,
        customer_id smallint NOT NULL,
        staff_id smallint NOT NULL,
        rental_id integer NOT NULL,
        amount numeric(5,2) NOT NULL,
        payment_date timestamp WITHOUT TIME ZONE NOT NULL,
        CONSTRAINT payment_pkey PRIMARY KEY (payment_id),
        CONSTRAINT payment_customer_id_fkey FOREIGN KEY (customer_id) REFERENCES public.customer (customer_id)
 MATCH SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT payment_rental_id_fkey FOREIGN KEY (rental_id) REFERENCES public.rental (rental_id) MATCH SIMPLE
 ON DELETE SET NULL ON UPDATE CASCADE,
        CONSTRAINT payment_staff_id_fkey FOREIGN KEY (staff_id) REFERENCES public.staff (staff_id) MATCH SIMPLE ON
 DELETE RESTRICT ON UPDATE CASCADE
 ) TABLESPACE pg_default;

 CREATE INDEX IF NOT EXISTS idx_fk_customer_id
  ON ONLY public.payment USING BTREE (customer_id pg_catalog.int2_ops ASC NULLS LAST) ;

 CREATE INDEX IF NOT EXISTS idx_fk_rental_id
  ON ONLY public.payment USING BTREE (rental_id pg_catalog.int4_ops ASC NULLS LAST) ;

 CREATE INDEX IF NOT EXISTS idx_fk_staff_id
  ON ONLY public.payment USING BTREE (staff_id pg_catalog.int2_ops ASC NULLS LAST) ;
```

```
CREATE UNIQUE INDEX IF NOT EXISTS payment_pkey
 ON ONLY public.payment USING BTREE (payment_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE TABLE IF NOT EXISTS public.store (
     store_id integer DEFAULT nextval('store_store_id_seq'::regclass) NOT NULL,
     manager_staff_id smallint NOT NULL,
     address_id smallint NOT NULL,
     last_update timestamp WITHOUT TIME ZONE DEFAULT E'now()' NOT NULL,
     CONSTRAINT store_pkey PRIMARY KEY (store_id),
     CONSTRAINT store_address_id_fkey FOREIGN KEY (address_id) REFERENCES public.address (address_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE,
     CONSTRAINT store_manager_staff_id_fkey FOREIGN KEY (manager_staff_id) REFERENCES public.staff (staff_id) MATCH
SIMPLE ON DELETE RESTRICT ON UPDATE CASCADE
) TABLESPACE pg_default;

CREATE TRIGGER last_updated BEFORE UPDATE
     ON public.store
     FOR EACH ROW
     EXECUTE FUNCTION last_updated();

CREATE UNIQUE INDEX IF NOT EXISTS idx_unq_manager_staff_id
 ON ONLY public.store USING BTREE (manager_staff_id pg_catalog.int2_ops ASC NULLS LAST) ;

CREATE UNIQUE INDEX IF NOT EXISTS store_pkey
 ON ONLY public.store USING BTREE (store_id pg_catalog.int4_ops ASC NULLS LAST) ;

CREATE VIEW public.actor_info
AS SELECT a.actor_id,
        a.first_name,
        a.last_name,
        group_concat(DISTINCT (((c.name)::text || ': '::text) || ( SELECT group_concat((f.title)::text) AS group_concat
             FROM ((film f
               JOIN film_category fc_1 ON ((f.film_id = fc_1.film_id)))
               JOIN film_actor fa_1 ON ((f.film_id = fa_1.film_id)))
            WHERE ((fc_1.category_id = c.category_id) AND (fa_1.actor_id = a.actor_id))
            GROUP BY fa_1.actor_id))) AS film_info
      FROM (((actor a
        LEFT JOIN film_actor fa ON ((a.actor_id = fa.actor_id)))
        LEFT JOIN film_category fc ON ((fa.film_id = fc.film_id)))
        LEFT JOIN category c ON ((fc.category_id = c.category_id)))
     GROUP BY a.actor_id, a.first_name, a.last_name;

CREATE VIEW public.customer_list
AS SELECT cu.customer_id AS id,
        (((cu.first_name)::text || ' '::text) || (cu.last_name)::text) AS name,
        a.address,
        a.postal_code AS "zip code",
        a.phone,
        city.city,
        country.country,
          CASE
            WHEN cu.activebool THEN 'active'::text
            ELSE ''::text
          END AS notes,
        cu.store_id AS sid
      FROM (((customer cu
        JOIN address a ON ((cu.address_id = a.address_id)))
        JOIN city ON ((a.city_id = city.city_id)))
        JOIN country ON ((city.country_id = country.country_id)));
```

```
CREATE VIEW public.film_list
AS SELECT film.film_id AS fid,
        film.title,
        film.description,
        category.name AS category,
        film.rental_rate AS price,
        film.length,
        film.rating,
        group_concat(((((actor.first_name)::text || ' '::text) || (actor.last_name)::text)) AS actors
       FROM (((((category
         LEFT JOIN film_category ON ((category.category_id = film_category.category_id)))
         LEFT JOIN film ON ((film_category.film_id = film.film_id)))
         JOIN film_actor ON ((film.film_id = film_actor.film_id)))
         JOIN actor ON ((film_actor.actor_id = actor.actor_id)))
      GROUP BY film.film_id, film.title, film.description, category.name, film.rental_rate, film.length, film.rating;


CREATE VIEW public.nicer_but_slower_film_list
AS SELECT film.film_id AS fid,
        film.title,
        film.description,
        category.name AS category,
        film.rental_rate AS price,
        film.length,
        film.rating,
        group_concat(((((upper("substring"((actor.first_name)::text, 1, 1)) || lower("substring"((actor.first_name)::text, 2))) ||
upper("substring"((actor.last_name)::text, 1, 1))) || lower("substring"((actor.last_name)::text, 2)))) AS actors
        FROM (((((category
         LEFT JOIN film_category ON ((category.category_id = film_category.category_id)))
         LEFT JOIN film ON ((film_category.film_id = film.film_id)))
         JOIN film_actor ON ((film.film_id = film_actor.film_id)))
         JOIN actor ON ((film_actor.actor_id = actor.actor_id)))
      GROUP BY film.film_id, film.title, film.description, category.name, film.rental_rate, film.length, film.rating;


CREATE VIEW public.sales_by_film_category
AS SELECT c.name AS category,
        sum(p.amount) AS total_sales
       FROM (((((payment p
         JOIN rental r ON ((p.rental_id = r.rental_id)))
         JOIN inventory i ON ((r.inventory_id = i.inventory_id)))
         JOIN film f ON ((i.film_id = f.film_id)))
         JOIN film_category fc ON ((f.film_id = fc.film_id)))
         JOIN category c ON ((fc.category_id = c.category_id)))
      GROUP BY c.name
      ORDER BY (sum(p.amount)) DESC;


CREATE VIEW public.sales_by_store
AS SELECT (((c.city)::text || ','::text) || (cy.country)::text) AS store,
        (((m.first_name)::text || ' '::text) || (m.last_name)::text) AS manager,
        sum(p.amount) AS total_sales
       FROM ((((((((payment p
         JOIN rental r ON ((p.rental_id = r.rental_id)))
         JOIN inventory i ON ((r.inventory_id = i.inventory_id)))
         JOIN store s ON ((i.store_id = s.store_id)))
         JOIN address a ON ((s.address_id = a.address_id)))
         JOIN city c ON ((a.city_id = c.city_id)))
         JOIN country cy ON ((c.country_id = cy.country_id)))
         JOIN staff m ON ((s.manager_staff_id = m.staff_id)))
      GROUP BY cy.country, c.city, s.store_id, m.first_name, m.last_name
      ORDER BY cy.country, c.city;


CREATE VIEW public.staff_list
```

```
AS SELECT s.staff_id AS id,
        (((s.first_name)::text || ' '::text) || (s.last_name)::text) AS name,
        a.address,
        a.postal_code AS "zip code",
        a.phone,
        city.city,
        country.country,
        s.store_id AS sid
    FROM (((staff s
     JOIN address a ON ((s.address_id = a.address_id)))
     JOIN city ON ((a.city_id = city.city_id)))
     JOIN country ON ((city.country_id = country.country_id)));
```

3. RELATIONSHIPS

3.1 Relationship customer_address_id_fkey

3.1.1 customer_address_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
address	address_id

CHILD TABLE	CHILD COLUMN
customer	address_id
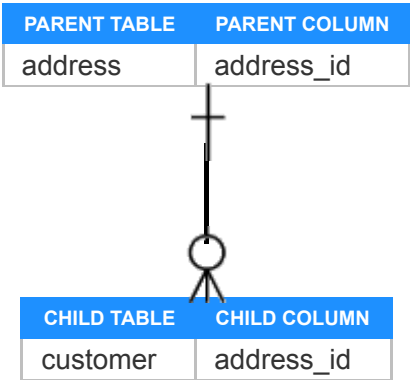
3.1.2 customer_address_id_fkey Properties

PROPERTY	VALUE
Name	customer_address_id_fkey
Description	
Parent Table	address
Parent Column	address_id
Parent Cardinality	1
Child Table	customer
Child Column	address_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.2 Relationship film_actor_actor_id_fkey

3.2.1 film_actor_actor_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
actor	actor_id

CHILD TABLE	CHILD COLUMN
film_actor	actor_id
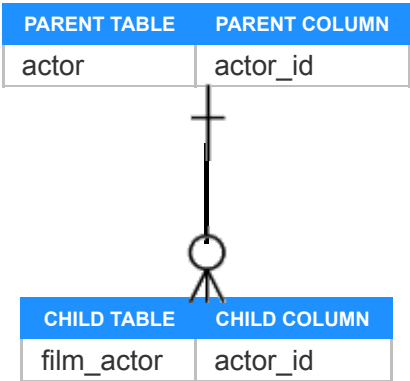
3.2.2 film_actor_actor_id_fkey Properties

PROPERTY	VALUE
Name	film_actor_actor_id_fkey
Description	
Parent Table	actor
Parent Column	actor_id
Parent Cardinality	1
Child Table	film_actor
Child Column	actor_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.3 Relationship film_actor_film_id_fkey

3.3.1 film_actor_film_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
film	film_id

CHILD TABLE	CHILD COLUMN
film_actor	film_id
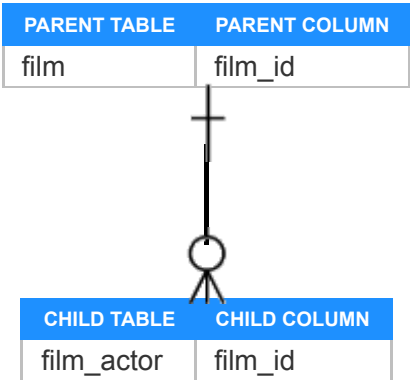
3.3.2 film_actor_film_id_fkey Properties

PROPERTY	VALUE
Name	film_actor_film_id_fkey
Description	
Parent Table	film
Parent Column	film_id
Parent Cardinality	1
Child Table	film_actor
Child Column	film_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.4 Relationship film_category_category_id_fkey

3.4.1 film_category_category_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
category	category_id

CHILD TABLE	CHILD COLUMN
film_category	category_id
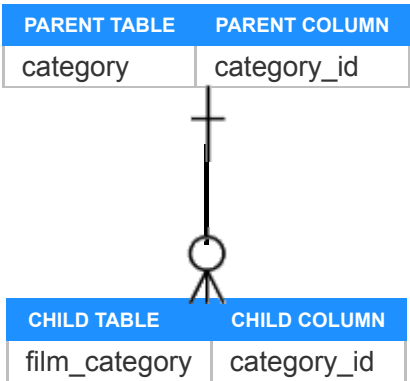
3.4.2 film_category_category_id_fkey Properties

PROPERTY	VALUE
Name	film_category_category_id_fkey
Description	
Parent Table	category
Parent Column	category_id
Parent Cardinality	1
Child Table	film_category
Child Column	category_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.5 Relationship film_category_film_id_fkey

3.5.1 film_category_film_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
film	film_id

CHILD TABLE	CHILD COLUMN
film_category	film_id
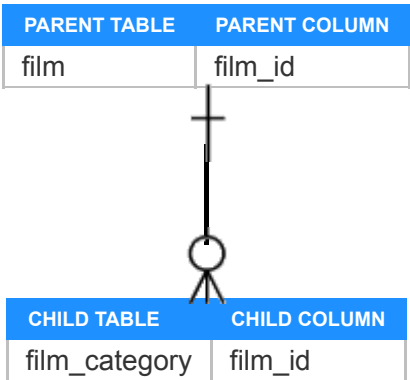
3.5.2 film_category_film_id_fkey Properties

PROPERTY	VALUE
Name	film_category_film_id_fkey
Description	
Parent Table	film
Parent Column	film_id
Parent Cardinality	1
Child Table	film_category
Child Column	film_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.6 Relationship film_language_id_fkey

3.6.1 film_language_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
language	language_id

CHILD TABLE	CHILD COLUMN
film	language_id
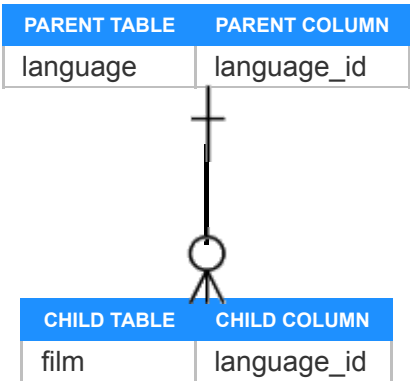
3.6.2 film_language_id_fkey Properties

PROPERTY	VALUE
Name	film_language_id_fkey
Description	
Parent Table	language
Parent Column	language_id
Parent Cardinality	1
Child Table	film
Child Column	language_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.7 Relationship fk_address_city

3.7.1 fk_address_city Diagram

PARENT TABLE	PARENT COLUMN
city	city_id

CHILD TABLE	CHILD COLUMN
address	city_id
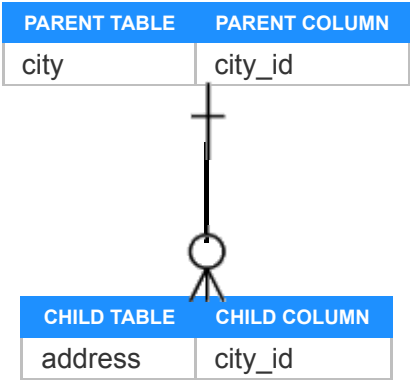
3.7.2 fk_address_city Properties

PROPERTY	VALUE
Name	fk_address_city
Description	
Parent Table	city
Parent Column	city_id
Parent Cardinality	1
Child Table	address
Child Column	city_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	NO ACTION

3.8 Relationship fk_city

3.8.1 fk_city Diagram

PARENT TABLE	PARENT COLUMN
country	country_id

CHILD TABLE	CHILD COLUMN
city	country_id
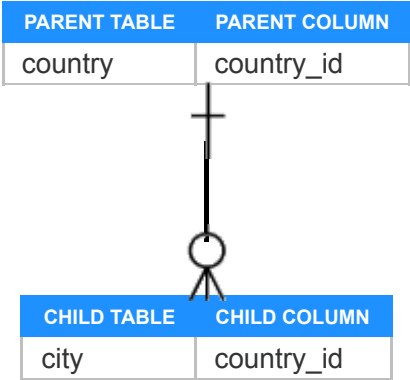
3.8.2 fk_city Properties

PROPERTY	VALUE
Name	fk_city
Description	
Parent Table	country
Parent Column	country_id
Parent Cardinality	1
Child Table	city
Child Column	country_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	NO ACTION

3.9 Relationship inventory_film_id_fkey

3.9.1 inventory_film_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
film	film_id

CHILD TABLE	CHILD COLUMN
inventory	film_id
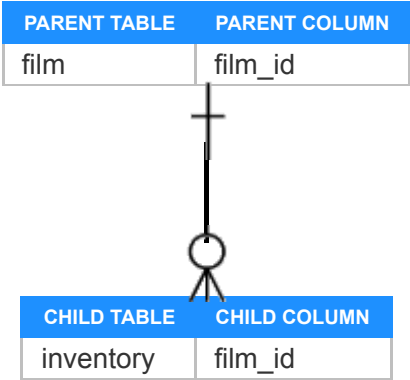
3.9.2 inventory_film_id_fkey Properties

PROPERTY	VALUE
Name	inventory_film_id_fkey
Description	
Parent Table	film
Parent Column	film_id
Parent Cardinality	1
Child Table	inventory
Child Column	film_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.10 Relationship payment_customer_id_fkey

3.10.1 payment_customer_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
customer	customer_id

CHILD TABLE	CHILD COLUMN
payment	customer_id
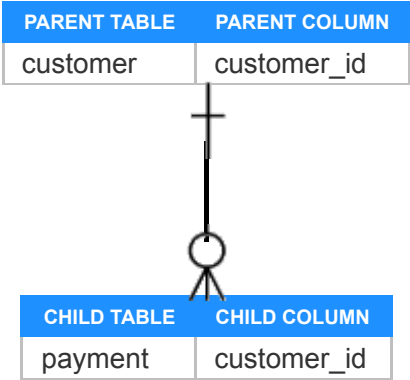
3.10.2 payment_customer_id_fkey Properties

PROPERTY	VALUE
Name	payment_customer_id_fkey
Description	
Parent Table	customer
Parent Column	customer_id
Parent Cardinality	1
Child Table	payment
Child Column	customer_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.11 Relationship payment_rental_id_fkey

3.11.1 payment_rental_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
rental	rental_id

CHILD TABLE	CHILD COLUMN
payment	rental_id
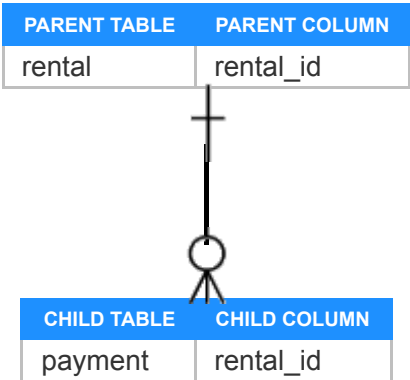
3.11.2 payment_rental_id_fkey Properties

PROPERTY	VALUE
Name	payment_rental_id_fkey
Description	
Parent Table	rental
Parent Column	rental_id
Parent Cardinality	1
Child Table	payment
Child Column	rental_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	SET NULL

3.12 Relationship payment_staff_id_fkey

3.12.1 payment_staff_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
staff	staff_id

CHILD TABLE	CHILD COLUMN
payment	staff_id
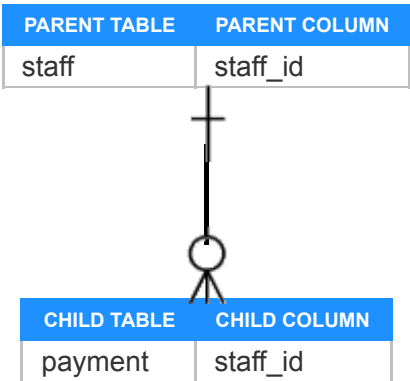
3.12.2 payment_staff_id_fkey Properties

PROPERTY	VALUE
Name	payment_staff_id_fkey
Description	
Parent Table	staff
Parent Column	staff_id
Parent Cardinality	1
Child Table	payment
Child Column	staff_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.13 Relationship rental_customer_id_fkey

3.13.1 rental_customer_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
customer	customer_id

CHILD TABLE	CHILD COLUMN
rental	customer_id
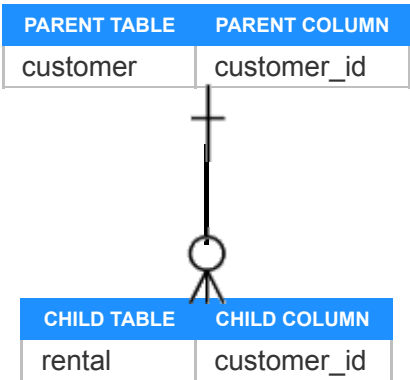
3.13.2 rental_customer_id_fkey Properties

PROPERTY	VALUE
Name	rental_customer_id_fkey
Description	
Parent Table	customer
Parent Column	customer_id
Parent Cardinality	1
Child Table	rental
Child Column	customer_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.14 Relationship rental_inventory_id_fkey

### 3.14.1 rental_inventory_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
inventory	inventory_id

CHILD TABLE	CHILD COLUMN
rental	inventory_id
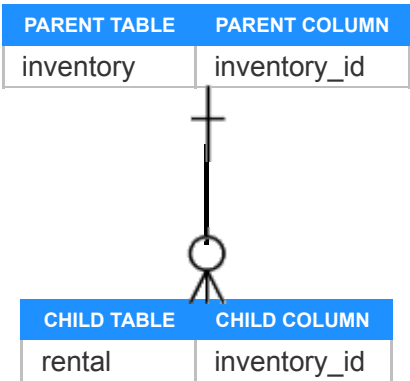
3.14.2 rental_inventory_id_fkey Properties

PROPERTY	VALUE
Name	rental_inventory_id_fkey
Description	
Parent Table	inventory
Parent Column	inventory_id
Parent Cardinality	1
Child Table	rental
Child Column	inventory_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.15 Relationship rental_staff_id_key

3.15.1 rental_staff_id_key Diagram

PARENT TABLE	PARENT COLUMN
staff	staff_id

CHILD TABLE	CHILD COLUMN
rental	staff_id
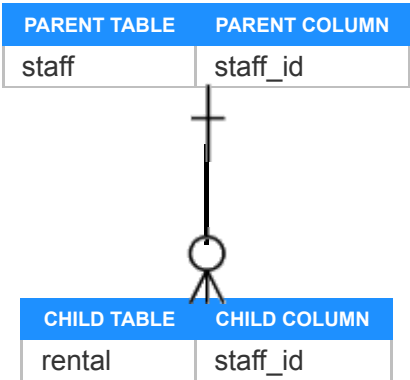
3.15.2 rental_staff_id_key Properties

PROPERTY	VALUE
Name	rental_staff_id_key
Description	
Parent Table	staff
Parent Column	staff_id
Parent Cardinality	1
Child Table	rental
Child Column	staff_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	NO ACTION

3.16 Relationship staff_address_id_fkey

3.16.1 staff_address_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
address	address_id

CHILD TABLE	CHILD COLUMN
staff	address_id
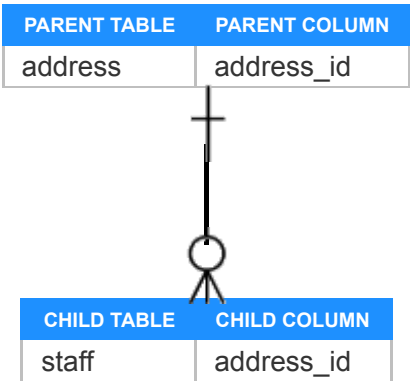
3.16.2 staff_address_id_fkey Properties

PROPERTY	VALUE
Name	staff_address_id_fkey
Description	
Parent Table	address
Parent Column	address_id
Parent Cardinality	1
Child Table	staff
Child Column	address_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.17 Relationship store_address_id_fkey

3.17.1 store_address_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
address	address_id

CHILD TABLE	CHILD COLUMN
store	address_id
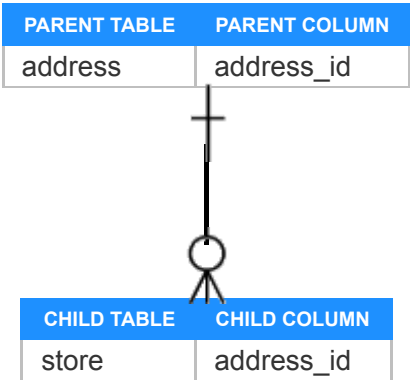
3.17.2 store_address_id_fkey Properties

PROPERTY	VALUE
Name	store_address_id_fkey
Description	
Parent Table	address
Parent Column	address_id
Parent Cardinality	1
Child Table	store
Child Column	address_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT

3.18 Relationship store_manager_staff_id_fkey

3.18.1 store_manager_staff_id_fkey Diagram

PARENT TABLE	PARENT COLUMN
staff	staff_id

CHILD TABLE	CHILD COLUMN
store	manager_staff_id
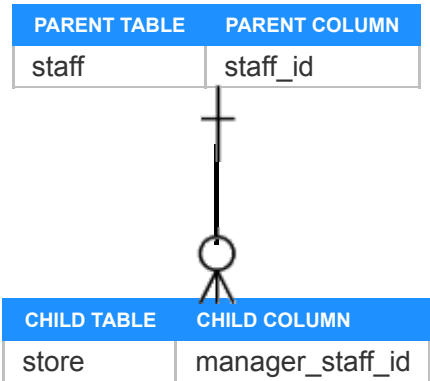
3.18.2 store_manager_staff_id_fkey Properties

PROPERTY	VALUE
Name	store_manager_staff_id_fkey
Description	
Parent Table	staff
Parent Column	staff_id
Parent Cardinality	1
Child Table	store
Child Column	manager_staff_id
Child Cardinality	0..n
Comments	
relationshipOnDelete	RESTRICT