



# وظيفة قواعد المعطيات المتقدمة

إشراف:

م. عبد البديع مراد

م. أبو الخير الصوص



إعداد الطلاب:

محمد مؤيد يونس       عمار رفاعي

نور الدين الطيلوني       أيهم الرفاعي

# الفهرس

## القسم الأول: أوراكل ..... 3

الطلب الأول: تعريف Tablespace باسم homeworkts حجمه 400 MB مؤلف من أربع ملفات معطيات Datafile.

الطلب الثاني: تعريف Profile باسم homeworkpf يحدد من خلاله السماح لمستخدم واحد الاتصال بقاعدة المطعيات بشرط أن تكون مدة الاتصال الفعال ساعة ومرة الاتصال الغير فعالة عشرة دقائق وضرورة تغيير كلمة السر كل سبعة أيام وحجم الذاكرة 50 كيلوبايت.

الطلب الثالث: تعريف حساب User جديد في قاعدة المطعيات باسم homeworku مرتبطة بال Tablespace homeworkts ونحوه homeworkpf.

الطلب الرابع: إعطاء كافة الصالحيات لهذا المستخدم من خلال Role واحدة، وأن يكون له صالحيات قراءة من جدول departments وتعديل وتحذيف بيانات جدول employees من حساب HR.

الطلب الخامس: إجراء نسخة احتياطية للحساب homeworku دون أخذ بيانات الجداول، وعرض ملف LOG.

## إنشاء قاعد المطعيات (تأجير الأفلام) ..... 8

8..... Language

8..... Category

8..... Actor

8..... Film

9..... Film\_Actor

9..... Film\_Category

9..... City

9..... Country

9..... Customer

9..... Address

10..... Inventory

10..... Store

10..... Rental

10.....	Staff
11.....	Payment
12.....	Database Diagram

## القسم الثاني: قواعد المعطيات الفعالة و PL\_SQL

13.....	الطلب الأول: إضافة الأفلام تم فقط يوم الخميس بين الساعة السادسة والثانية صباحا، ويمنع حذف أي فلم.
14.....	الطلب الثاني: قيمة الإيجار في جدول الدفعات يجب أن تكون متساوية لقيمة تكلفة الإيجار من جدول الأفلام كافة أيام الأسبوع، إلا في حال كان يوم الإيجار سبت أو أحد تضافر قيمة 15% على تكلفة الإيجار.
15.....	الطلب الثالث: منع تأجير نفس الفيلم لنفس الزبون أكثر من مرة بالشهر.
16.....	الطلب الرابع: بناء سجل متابعة لعمليات الإضافة والتعديل لجدول الحجوزات في جدول رديف (يجب بناء الجدول) يسجل قيم جميع الحقول قبل وبعد تنفيذ العملية مع نوع وتاريخ وזמן العملية.
17.....	الطلب الخامس: تسجيل زمن دخول وخروج أي مستخدم لقاعدة المعطيات في جدول مناسب ضمن حساب System يستثنى من ذلك حساب homeworku

## القسم الثالث: قواعد المعطيات متعددة الأبعاد

22.....	Customer_Dim
22.....	Film_Dim
22.....	Rental_Fact
22.....	Time_Dim
23.....	الطلب الأول: بناء كافة الأبعاد مع مراعاة عملية التعديل التاريخية لبعد الزبون، وتحديد جميع الهرميات المناسبة لكل بعد.
24.....	الطلب الثاني: تجزئة جدول Fact بالطريقة التي تراها مناسبة وتعديل ذلك، هل من الممكن أن تكون هذه التجزئة مركبة وضيق ذلك.
25.....	الطلب الثالث: بناء MATERIALIZED VIEW يتضمن اسم الفيلم واسم الزبون وفترة الحجز، تتحدد بيانات هذا المنظور بشكل تراكمي.
26.....	الطلب الرابع: عرض قائمة بقيم حجوزات الأفلام على مستوى الشهر والبلد والمدينة بكافة الاحتمالات الممكنة.
27.....	الطلب الخامس: أوجد ترتيب كل سنة وكل فيلم ضمن هذه السنة حسب قيمة التأجير.
28.....	الطلب السادس: يريد مدير الشركة أن يعرف ما هي العلاقة ما بين مدينة الزبون وفترة الأفلام ولغة الفيلم لكي يستخدم هذه المعلومات في الإعلانات الموجهة، اقترح طريقة مناسبة لمعرفة هذه العلاقة ثمنفذها باستخدام weka.

## القسم الأول: أوراكل

الطلب الأول: تعريف Tablespace باسم homeworks حجمه 400 MB مؤلف من أربع ملفات Datafile.

```
CREATE TABLESPACE homeworks
  DATAFILE
    'C:\app\Ali\product\11.2.0\dbhome_1\oradata\homework\homeworkfile1.dbf' size 100M,
    'C:\app\Ali\product\11.2.0\dbhome_1\oradata\homework\homeworkfile2.dbf' size 100M,
    'C:\app\Ali\product\11.2.0\dbhome_1\oradata\homework\homeworkfile3.dbf' size 100M,
    'C:\app\Ali\product\11.2.0\dbhome_1\oradata\homework\homeworkfile4.dbf' size 100M
  LOGGING
  Extent Management local
  SEGMENT SPACE MANAGEMENT AUTO;
```

TABLESPACE HOMWORKTS created.

```
SELECT
  a.tablespace_name,
  b.file_name
FROM
  dba tablespaces a
  JOIN dba data_files b ON ( a.tablespace_name = b.tablespace_name )
WHERE
  a.tablespace_name = 'HOMWORKTS'
```

وبعد إنشاء ال TABLESPACE نتأكد من وجوده ومن الملفات ال Datafiles المرتبطة به عن طريق الاستعلام التالي:

TABLESPACE_NAME	FILE_NAME
HOMEWORKTS	C:\APP\ALI\PRODUCT\11.2.0\DBHOME_1\ORADATA\HOMEWORK\HOMEWORKFILE1.DBF
HOMEWORKTS	C:\APP\ALI\PRODUCT\11.2.0\DBHOME_1\ORADATA\HOMEWORK\HOMEWORKFILE2.DBF
HOMEWORKTS	C:\APP\ALI\PRODUCT\11.2.0\DBHOME_1\ORADATA\HOMEWORK\HOMEWORKFILE3.DBF
HOMEWORKTS	C:\APP\ALI\PRODUCT\11.2.0\DBHOME_1\ORADATA\HOMEWORK\HOMEWORKFILE4.DBF

ونجد أيضاً الملفات في المسار المذكور:

	Name	Date modified	Type	Size
0	HOMEWORKFILE1.DBF	12/2/2023 10:45 A...	DBF File	102,408 KB
ads	HOMEWORKFILE2.DBF	12/2/2023 10:45 A...	DBF File	102,408 KB
Places	HOMEWORKFILE3.DBF	12/2/2023 10:45 A...	DBF File	102,408 KB
	HOMEWORKFILE4.DBF	12/2/2023 10:45 A...	DBF File	102,408 KB

الطلب الثاني: تعريف Profile باسم homeworkpf يحدد من خلاله السماح لمستخدم واحد الاتصال بقاعدة المعطيات بشرط أن تكون مدة الاتصال الفعال ساعة ومدة الاتصال الغير فعالة عشرة دقائق وضرورة تغيير كلمة السر كل سبعة أيام وحجم الذاكرة 50 كيلوبايت.

```
CREATE PROFILE homeworkpf LIMIT
SESSIONS_PER_USER 1
CONNECT_TIME 60
IDLE_TIME 10
PASSWORD_LIFE_TIME 7
PRIVATE_SGA 50K;|
```

Script Output | Task completed in 0.095 seconds

Profile HOMEWORKPF created.

وتأكد من إنشاء Profile والخصائص المسندة له بالاستعلام التالي:

```
SELECT
*
FROM
dba_profiles
ORDER BY
profile;
```

Query Result | All Rows Fetched: 48 in 0.017 seconds

PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT
HOMEWORKPF	PASSWORD_LOCK_TIME	PASSWORD	DEFAULT
HOMEWORKPF	PASSWORD_GRACE_TIME	PASSWORD	DEFAULT
HOMEWORKPF	PASSWORD_VERIFY_FUNCTION	PASSWORD	DEFAULT
HOMEWORKPF	PASSWORD_REUSE_MAX	PASSWORD	DEFAULT
HOMEWORKPF	PASSWORD_REUSE_TIME	PASSWORD	DEFAULT
HOMEWORKPF	PASSWORD_LIFE_TIME	PASSWORD	7
HOMEWORKPF	FAILED_LOGIN_ATTEMPTS	PASSWORD	DEFAULT
HOMEWORKPF	PRIVATE_SGA	KERNEL	51200
HOMEWORKPF	CONNECT_TIME	KERNEL	60
HOMEWORKPF	IDLE_TIME	KERNEL	10
HOMEWORKPF	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT
HOMEWORKPF	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT
HOMEWORKPF	CPU_PER_CALL	KERNEL	DEFAULT
HOMEWORKPF	CPU_PER_SESSION	KERNEL	DEFAULT
HOMEWORKPF	SESSIONS_PER_USER	KERNEL	1
HOMEWORKPF	COMPOSITE_LIMIT	KERNEL	DEFAULT

الطلب الثالث: تعریف حساب User جديد في قاعدة المعطیات باسم homeworku مرتبطة بـ Profile homeworkpf و منحه homeworkts باسم Tablespace.

```

CREATE USER homeworku
IDENTIFIED BY homework
PROFILE homeworkpf
DEFAULT TABLESPACE homeworkts
QUOTA UNLIMITED ON homeworkts
ACCOUNT UNLOCK;

```

Script Output | Task completed in 0.029 seconds

User HOMEWORKU created.

نتأكد من انشاء الـUser والخصائص المسندة له بالاستعلام التالي:

```

SELECT
    username, user_id,
    account_status, expiry_date,
    default_tablespace, temporary_tablespace,
    created, profile
FROM
    dba_users;

```

Query Result | SQL | All Rows Fetched: 38 in 0.026 seconds

USERNAME	USER_ID	ACCOUNT_STATUS	EXPIRY_DATE	DEFAULT_TABLESPACE	TEMPORARY_TABLESPACE	CREATED	PROFILE
1 SYSTEM	5 OPEN	05-MAY-24	SYSTEM	TEMP		09-DEC-10	DEFAULT
2 SYS	0 OPEN	29-MAY-22	SYSTEM	TEMP		09-DEC-10	DEFAULT
3 MGMT_VIEW	73 OPEN	29-MAY-22	SYSTEM	TEMP		09-DEC-10	DEFAULT
4 SMITH	90 OPEN	29-MAY-22	USERS	TEMP		30-NOV-21	DEFAULT
5 HR	84 OPEN	29-MAY-24	USERS	TEMP		30-NOV-21	DEFAULT
6 HOMEWORKU	91 OPEN	09-DEC-23	HOMWORKTS	TEMP		02-DEC-23	HOMWORKPF

الطلب الرابع: إعطاء كافة الصالحيات لهذا المستخدم من خلال Role واحدة، وأن يكون له صالحيات قراءة من جدول departments وتعديل وحذف بيانات جدول employees من حساب HR.

```

CREATE ROLE homeworkro;
GRANT connect, resource TO homeworkro;
GRANT SELECT ON hr.departments TO homeworkro;
GRANT UPDATE, DELETE ON hr.employees TO homeworkro;

GRANT homeworkro TO homeworku;

```

Script Output | Task completed in 0.033 seconds

Role HOMEWORKRO created.

Grant succeeded.

Grant succeeded.

Grant succeeded.

Grant succeeded.

```

SELECT
  *
FROM
  dba_role_privs
WHERE
  grantee = 'HOMEWORKU';

```

Query... | SQL | All Rows Fetched: 1 in 0.038 seconds

GRANTEE	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE
1 HOMEWORKU	HOMEROORKO	NO	YES

قمت بإنشاء Role باسم homeworkro  
وقمت بإعطائه كافة الصالحيات  
اللازمة والمطلوبة في نص السؤال وفي  
السطر الأخير قمت بإسناد هذا الـ Role  
للمستخدم الذي أنشأناه سابقاً ونتأكّد  
من الاستناد بالاستعلام التالي:

ولرؤية الصالحيات التي استندناها لـ Role نقوم بالاستعلامات التالية

```

SELECT
  *
FROM
  ROLE_ROLE_PRIVS
WHERE
  role = 'HOMEROORKO';

```

Query Result | SQL | All Rows Fetched: 2 in 0.006 seconds

ROLE	GRANTED_ROLE	ADMIN_OPTION
1 HOMEROORKO	RESOURCE	NO
2 HOMEROORKO	CONNECT	NO

```

SELECT
  *
FROM
  ROLE_TAB_PRIVS
WHERE
  role = 'HOMEROORKO';

```

Query Result | SQL | All Rows Fetched: 3 in 0.057 seconds

ROLE	OWNER	TABLE_NAME	COLUMN_NAME	PRIVILEGE	GRANTABLE
1 HOMEROORKO	HR	DEPARTMENTS	(null)	SELECT	NO
2 HOMEROORKO	HR	EMPLOYEES	(null)	UPDATE	NO
3 HOMEROORKO	HR	EMPLOYEES	(null)	DELETE	NO

إذا قمنا بتجربة تنفيذ الاستعلامات على جدول employees and departments من حساب hr. وضمن الـ User homeworku سينجح من أجل الأقسام ويفشل من أجل الموظفين كالتالي:

```

SELECT
*
FROM
hr.departments;

```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	Administration	200	1700
2	Marketing	201	1800
3	Purchasing	114	1700
4	Human Resources	203	2400
5	Shipping	121	1500

```

SELECT
*
FROM
HR.employees;

```

Query Result x  
SQL | Executing:SELECT \* FROM HR.employees in 0 seconds

ORA-01031: insufficient privileges  
01031. 00000 - "insufficient privileges"  
\*Cause: An attempt was made to perform a database operation without the necessary privileges.  
\*Action: Ask your database administrator or designated security administrator to grant you the necessary privileges  
Error at Line: 4 Column: 8

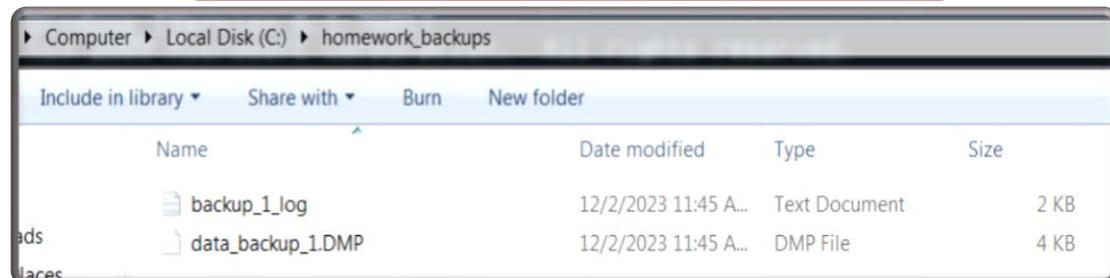
الطلب الخامس: إجراء نسخة احتياطية للحساب homeworku دون أخذ بيانات الجداول،  
وعرض ملف LOG.

```

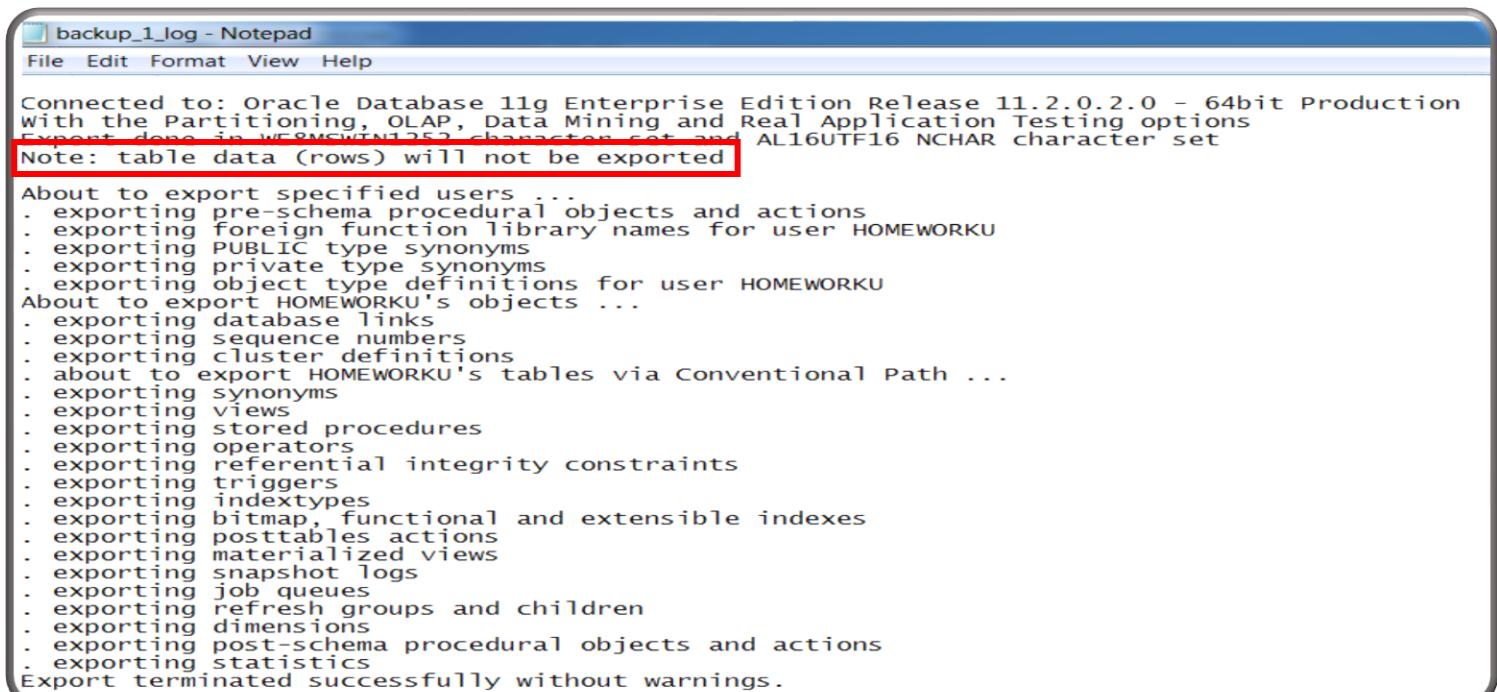
EXP.EXE USERID='sys/Passw0rd as sysdba'
OWNER="homeworku"
FILE=c:\homework_backups\data_backup_1.DMP
COMPRESS=Y GRANTS=Y INDEXES=Y
LOG=c:\homework_backups\backup_1_log.LOG
ROWS=N CONSTRAINTS=Y

```

نقوم بكتابة التعليمية التالية في الـ  
CMD ضمن مجلد BIN الموجود في  
ـOracle مجلد قاعدة المعطيات



فيتم انشاء النسخة  
الاحتياطية ضمن  
المجلد المحدد:  
ـ والـ Log file كالتالي:



## إنشاء قاعد المعطيات (تأجير الأفلام)

### Category

### Language

```
CREATE TABLE category (
    id          NUMBER(10) NOT NULL,
    name        VARCHAR2(25) NOT NULL,
    last_update TIMESTAMP,
    CONSTRAINT category_pk PRIMARY KEY (id)
);
```

```
CREATE TABLE language (
    id          NUMBER(10) NOT NULL,
    name        VARCHAR2(20) NOT NULL,
    last_update TIMESTAMP,
    CONSTRAINT language_pk PRIMARY KEY (id)
);
```

### Film

### Actor

```
CREATE TABLE film (
    id                  NUMBER(10) NOT NULL,
    title               VARCHAR2(255) NOT
NULL,
    description         VARCHAR2(255),
    release_year       NUMBER(4),
    rental_duration    NUMBER(10),
    rental_rate        NUMBER(19, 0) NOT
NULL,
    length              NUMBER(2),
    replacement_cost   NUMBER(19, 0),
    rating              NUMBER(10),
    last_update         TIMESTAMP,
    special_features   VARCHAR2(255),
    full_text           VARCHAR2(255),
    languageid          NUMBER(10),
    CONSTRAINT film_pk PRIMARY KEY ( id ),
    CONSTRAINT film_language_fk FOREIGN KEY
( languageid )
        REFERENCES language ( id )
        ON DELETE CASCADE
);
```

```
CREATE TABLE actor (
    id          NUMBER(10) NOT NULL,
    first_name  VARCHAR2(255) NOT NULL,
    last_name   VARCHAR2(255) NOT NULL,
    last_update TIMESTAMP,
    CONSTRAINT actor_pk PRIMARY KEY ( id )
);
```

**Film\_Category**

```
CREATE TABLE film_category (
    filmid      NUMBER(10) NOT NULL,
    categoryid  NUMBER(10) NOT NULL,
    last_update  TIMESTAMP,
    CONSTRAINT film_category_uk UNIQUE
(filmid, categoryid),
    CONSTRAINT film_fk FOREIGN KEY (filmid)
        REFERENCES film ( id )
        ON DELETE CASCADE,
    CONSTRAINT film_category_fk FOREIGN KEY
(categoryid)
        REFERENCES category ( id )
        ON DELETE CASCADE
);
```

**Film\_Actor**

```
CREATE TABLE film_actor (
    filmid      NUMBER(10) NOT NULL,
    actorid     NUMBER(10),
    last_update  TIMESTAMP,
    CONSTRAINT film_actor_uk UNIQUE
(filmid, actorid),
    CONSTRAINT film_film_actor_fk FOREIGN
KEY (filmid)
        REFERENCES film ( id )
        ON DELETE CASCADE,
    CONSTRAINT film_actor_fk FOREIGN KEY (
actorid )
        REFERENCES actor ( id )
        ON DELETE CASCADE);
```

**Country**

```
CREATE TABLE country (
    id          NUMBER(10) NOT NULL,
    country     VARCHAR2(50) NOT NULL,
    last_update  TIMESTAMP,
    CONSTRAINT country_pk PRIMARY KEY (id)
);
```

**City**

```
CREATE TABLE city (
    id          NUMBER(10) NOT NULL,
    city        VARCHAR2(50) NOT NULL,
    countryid   NUMBER(10) NOT NULL,
    last_update  TIMESTAMP,
    CONSTRAINT city_pk PRIMARY KEY ( id ),
    CONSTRAINT city_country_fk FOREIGN KEY
(countryid)
        REFERENCES country ( id )
        ON DELETE CASCADE
);
```

**Address**

```
CREATE TABLE address (
    id          NUMBER(10) NOT NULL,
    address     VARCHAR2(50) NOT NULL,
    address2    VARCHAR2(50),
    district    NUMBER(20),
    postal_code VARCHAR2(10),
```

**Customer**

```
CREATE TABLE customer (
    id          NUMBER(10) NOT NULL,
    first_name  VARCHAR2(255) NOT NULL,
    last_name   VARCHAR2(255) NOT NULL,
    email       VARCHAR2(50) NOT NULL,
    active      CHAR(1) NOT NULL,
```

```

phone      VARCHAR2(20),
cityid     NUMBER(10) NOT NULL,
last_update TIMESTAMP,
CONSTRAINT address_pk PRIMARY KEY ( id ),
CONSTRAINT address_city_fk FOREIGN KEY
( cityid )
    REFERENCES city ( id )
    ON DELETE CASCADE
);

```

```

addresscolumn NUMBER(10),
addressid    NUMBER(10) NOT NULL,
created_date TIMESTAMP NOT NULL,
last_update   TIMESTAMP,
CONSTRAINT customer_pk PRIMARY KEY ( id ),
CONSTRAINT customer_address_fk FOREIGN
KEY ( addressid )
    REFERENCES address ( id )
    ON DELETE CASCADE
);

```

**Store**

```

CREATE TABLE store (
    id          NUMBER(10) NOT NULL,
    addressid   NUMBER(10) NOT NULL,
    last_update  TIMESTAMP,
    CONSTRAINT store_pk PRIMARY KEY ( id ),
    CONSTRAINT store_address_fk FOREIGN KEY
( addressid )
    REFERENCES address ( id )
    ON DELETE CASCADE
);

```

**Inventory**

```

CREATE TABLE inventory (
    id          NUMBER(10) NOT NULL,
    filmid      NUMBER(10) NOT NULL,
    last_update  TIMESTAMP,
    CONSTRAINT inventory_pk PRIMARY KEY
(id),
    CONSTRAINT inventory_film_fk FOREIGN
KEY ( filmid )
    REFERENCES film ( id )
    ON DELETE CASCADE);

```

**Staff**

```

CREATE TABLE staff (
    id          NUMBER(10) NOT NULL,
    first_name  VARCHAR2(255) NOT NULL,
    last_name   VARCHAR2(255) NOT NULL,
    username    VARCHAR2(16) NOT NULL,
    email       VARCHAR2(50) NOT NULL,
    password    VARCHAR2(40) NOT NULL,
    active      CHAR(1) NOT NULL,
    pictureurl  VARCHAR2(80),
    addressid   NUMBER(10) NOT NULL,
    storeid     NUMBER(10) NOT NULL,
    last_update  TIMESTAMP,

```

```

CREATE TABLE rental (
    id          NUMBER(10) NOT NULL,
    rental_date TIMESTAMP NOT NULL,
    return_date TIMESTAMP,
    staffid     NUMBER(10),
    customerid  NUMBER(10),
    inventoryid NUMBER(10),
    last_update  TIMESTAMP,
    CONSTRAINT rental_pk PRIMARY KEY ( id ),
    CONSTRAINT rental_staff_fk FOREIGN KEY
( staffid )

```

```

CONSTRAINT staff_pk PRIMARY KEY ( id ),
CONSTRAINT staff_username_uk UNIQUE
(username),
CONSTRAINT staff_email_uk UNIQUE
(email),
CONSTRAINT staff_address_fk FOREIGN KEY
(addressid)
    REFERENCES address ( id )
    ON DELETE CASCADE,
CONSTRAINT staff_store_fk FOREIGN KEY
(storeid)
    REFERENCES store ( id )
    ON DELETE CASCADE
);

```

```

REFERENCES staff ( id )
    ON DELETE CASCADE,
CONSTRAINT rental_customer_fk FOREIGN
KEY ( customerid )
    REFERENCES customer ( id )
    ON DELETE CASCADE,
CONSTRAINT rental_inventory_fk FOREIGN
KEY ( inventoryid )
    REFERENCES inventory ( id )
    ON DELETE CASCADE
);

```

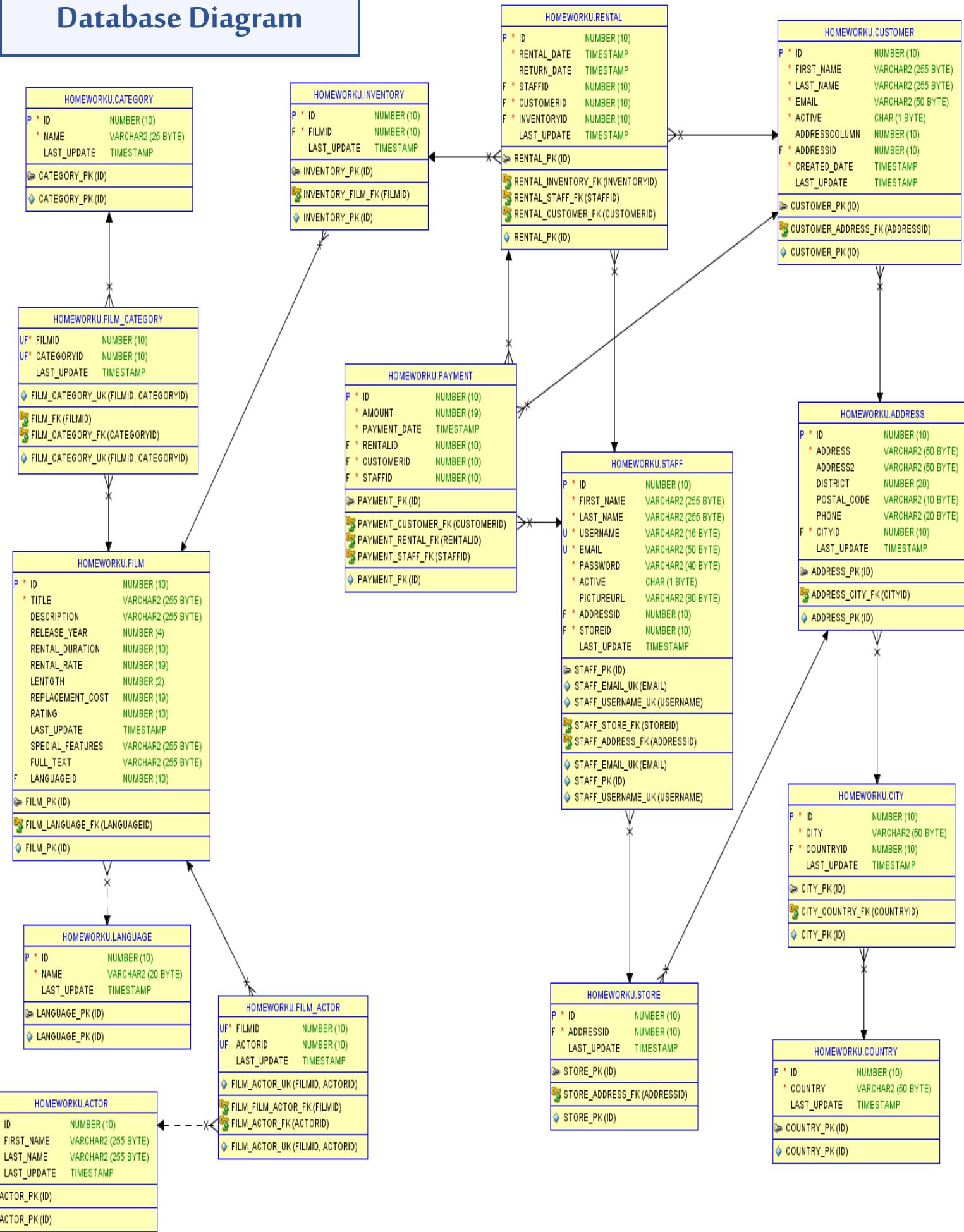
## Payment

```

CREATE TABLE payment (
    id          NUMBER(10) NOT NULL,
    amount      NUMBER(19, 0) NOT NULL,
    payment_date TIMESTAMP NOT NULL,
    rentalid    NUMBER(10) NOT NULL,
    customerid  NUMBER(10) NOT NULL,
    staffid     NUMBER(10) NOT NULL,
    CONSTRAINT payment_pk PRIMARY KEY ( id ),
    CONSTRAINT payment_rental_fk FOREIGN KEY ( rentalid )
        REFERENCES rental ( id )
        ON DELETE CASCADE,
    CONSTRAINT payment_customer_fk FOREIGN KEY ( customerid )
        REFERENCES customer ( id )
        ON DELETE CASCADE,
    CONSTRAINT payment_staff_fk FOREIGN KEY ( staffid )
        REFERENCES staff ( id )
        ON DELETE CASCADE
);

```

# Database Diagram



## القسم الثاني: قواعد المعطيات الفعالة و PL\_SQL

**الطلب الأول:** إضافة الأفلام تتم فقط يوم الخميس بين الساعة السادسة والثانية صباحا، ويمنع حذف أي فلم.

```

CREATE OR REPLACE TRIGGER film_insert_delete_trigger BEFORE
    INSERT OR DELETE ON film
BEGIN
    IF
        (to_char(sysdate, 'dy') != 'thu' AND inserting)
    OR
        to_char(sysdate, 'dy') = 'thu'
        AND
        (
            (to_char(sysdate, 'HH24') NOT BETWEEN 6 AND 8
                AND inserting)
            OR
            (to_char(sysdate, 'HH24') BETWEEN 6 AND 8 AND deleting)
        )
    THEN
        IF inserting THEN
            raise_application_error(-20500, 'Inserting only allowed
                On Thursday between 6 AM and 8 AM');
        ELSE
            raise_application_error(-20501, 'Deleting not allowed On
                Thursday between 6 AM and 8 AM');
        END IF;
    END IF;
END;

```

شرح الشروط الخاصة بالقادر السابق:

- إذا لم يكن اليوم هو الخميس وكانت العملية التي نقوم بها هي إضافة فيتم إعطاء خطأ.
- إذا كان اليوم هو الخميس وكانت العملية هي إضافة فلم ولم يكن الوقت بين السادسة والثانية صباحا فيتم رفع خطأ أو كان اليوم هو الخميس وكانت العملية هي حذف وكان الوقت بين السادسة والثانية صباحا فيتم رفع خطأ وما تبقى من أوقات وايام فتصبح فيها عملية الحذف.

نلاحظ اننا قمنا بعمل **ROW LEVEL TRIGGER** وليس **STATEMENT LEVEL TRIGGER** أي لم نذكر **FOR EACH ROW** في عملية الانشاء لأن الطلب هنا لا يهتم بما تحتويه كل عملية DML من **OLD OR NEW VALUES** ولا بعدهم انما **STATEMENT LEVEL TRIGGER** يمنع العمليات في أوقات محددة وهذه من الاستخدامات الشائعة لـ

نقوم بتجربة بعض العمليات كالتالي:

إضافة فيلم في يوم الخميس في الوقت المحدد:

إضافة فيلم في غير يوم الخميس او في يوم الخميس

لكن ليس في الوقت المحدد:

```
INSERT INTO film (
    id,
    title,
    rental_rate
) VALUES (
    1,
    'Anything',
    1900
);
```

Script Output x

Task completed in 0.099 seconds

1 row inserted.

```
INSERT INTO film (
    id,
    title,
    rental_rate
) VALUES (
    1,
    'Anything',
    1900
);
```

Script Output x | Task completed in 0.085 seconds

```
id,
title,
rental_rate
) VALUES (
    1,
    'Anything',
    1900
)
```

Error report -

ORA-20500: Inserting only allowed On Thursday between 6 AM and 8 AM  
ORA-06512: at "HOMWORKU.FILM\_INSERT\_DELETE\_TRIGGER", line 26

ORA-04088: error during execution of trigger 'HOMWORKU.FILM\_INSERT\_DELETE\_TRIGGER'

الطلب الثاني: قيمة الايجار في جدول الدفعات يجب أن تكون مساوية لقيمة تكلفة الايجار من جدول الافلام كافة أيام الأسبوع، الا في حال كان يوم الايجار سبت أو أحد تضاف قيمة 15% على تكلفة الايجار.

```
CREATE OR REPLACE TRIGGER payment_rental_film_trigger BEFORE
    INSERT ON payment
    FOR EACH ROW
DECLARE
    rate NUMBER(19,0);
    less_than EXCEPTION;
    great_than EXCEPTION;
BEGIN
    SELECT rental_rate INTO rate FROM film
    WHERE id = ( SELECT filmid FROM inventory
        WHERE id = ( SELECT inventoryid FROM rental
            WHERE id = :new.rentalid ) );
    IF to_char(sysdate, 'dy') IN ('sat','sun')
    THEN
```

```

rate := rate + rate * 0.15;
END IF;
IF :new.amount < rate THEN raise less_than;
ELSIF :new.amount > rate THEN raise great_than;
END IF;
EXCEPTION
WHEN less_than THEN
    raise_application_error(-20500,'Payment amount must be
as same as the Film''s Rental Rate which is: ' || rate
|| ' but you only enterd: ' || :new.amount);
WHEN great_than THEN
    raise_application_error(-20501,'Don''t thief people
the film''s rental rate is only: ' || rate
|| ' but you enterd: ' || :new.amount);
END;

```

قمنا بعمل استعلام بسيط لجلب سعر ايجار الفلم وذلك بالاستفادة من العلاقات بين الجداول وصولاً لجدول الأفلام وبعد ذلك تأكينا فيما ان كان اليوم هو سبت او أحد واضفنا 15% على سعر الايجار وان لم يكن كذلك سيبقى السعر هو نفسه سعر الايجار للفلم وقمنا برفع خطأ في حال كان اقل او أكثر لأن السعر يجب ان يكون متساوي.

تجربة بعض العمليات:

Worksheet | Query Builder

```

INSERT INTO payment (
  id, staffid, customerid,
  rentalid, payment_date, amount)
VALUES (
  1, 1, 1, 1, sysdate, 1900
);

```

Script Output x | Task completed in 0.138 seconds

Error starting at line : 1 in command -

```

INSERT INTO payment (
  id, staffid, customerid,
  rentalid, payment_date, amount)
VALUES (
  1, 1, 1, 1, sysdate, 1900
)
Error report -
ORA-20500: Payment amount must be as same as the Film's Rental Rate which is: 2185 but you only enterd: 1900
ORA-06512: at "HOMEWORKU.PAYMENT_RENTAL_FILM_TRIGGER", line 19
ORA-04088: error during execution of trigger 'HOMEWORKU.PAYMENT_RENTAL_FILM_TRIGGER'

```

Worksheet | Query Builder

```

INSERT INTO payment (
  id, staffid, customerid,
  rentalid, payment_date, amount)
VALUES (
  1, 1, 1, 1, sysdate, 2185
);

```

Script Output x | Task completed in 0.059 seconds

1 row inserted.

إضافة دفعه للإيجار بالسعر الموافق لسعر الإيجار الخاص بالفيلم ولكن  
في يومي السبت او الأحد:

```

    INSERT INTO payment (
        id, staffid, customerid,
        rentalid, payment_date, amount)
    VALUES (
        1, 1, 1, 1, sysdate, 1300
    );

```

إضافة دفعه للايجار بسعر اقل لسعر الايجار الخاص بالفلم في غير يومي

السبت او الاحد:

Script Output X | Task completed in 0.069 seconds

```

Error starting at line : 1 in command -
INSERT INTO payment (
    id, staffid, customerid,
    rentalid, payment_date, amount)
VALUES (
    1, 1, 1, 1, sysdate, 1300
)
Error report -
ORA-20500: Payment amount must be as same as the Film's Rental Rate which is: 1900 but you only enterd: 1300
ORA-06512: at "HOMEWORKU.PAYMENT_RENTAL_FILM_TRIGGER", line 19
ORA-04088: error during execution of trigger 'HOMEWORKU.PAYMENT_RENTAL_FILM_TRIGGER'

```

الطلب الثالث: منع تأجير نفس الفيلم لنفس الزبون أكثر من مرة بالشهر.

```

CREATE OR REPLACE TRIGGER rental_film_once_trigger BEFORE
    INSERT ON rental
    FOR EACH ROW
DECLARE
    selected_film NUMBER(10);
    count_of_rental_film_in_month number:=0;
    myexc EXCEPTION;
BEGIN
    SELECT filmid INTO selected_film FROM inventory WHERE
        id = :new.inventoryid;
    SELECT COUNT(*) INTO count_of_rental_film_in_month FROM rental
    WHERE customerid = :new.customerid AND inventoryid IN
        (SELECT id FROM inventory WHERE filmid = selected_film)
        AND to_char(sysdate, 'MM') = to_char(rental_date, 'MM');
    IF (count_of_rental_film_in_month > 0) THEN
        RAISE myexc;
    END IF;
EXCEPTION
    WHEN myexc THEN
        raise_application_error(-20500, 'You Can''t Rental
        the same Film Twice in a month');
END;

```

قمنا أولاً بجلب الفلم المراد حجزه ووضعناه ضمن متحول بالاستعلام الأول ثم بالاستعلام الثاني قمنا بجلب عدد الأفلام التي حجزها هذا العميل ولنفس الفلم أيا كان الـ `inventory` لهذا الفلم وضمن الشهر نفسه ثم اخترنا انه إذا كان أكبر من الصفر أي لديه حجز سابق لهذا الفلم ضمن الشهر نفسه فيعطي خطاً.

تجربة العمليات لذلك:

```

INSERT INTO rental (
    id, staffid, customerid,
    inventoryid, rental_date
)
VALUES (
    1, 1, 1, 1,
    to_timestamp(
        '2023-12-11 06:14:00.742000000',
        'YYYY-MM-DD HH24:MI:SS.FF'
    )
);
```

Script Output × | Task completed in 0.096 seconds

1 row inserted.

نقوم بإضافة إيجار جديد من أجل العميل صاحب الـ ID رقم 1 للفلم الموجود لا `inventoryid` رقم 1 (وهو الفلم الأول) فتتم العملية بنجاح:

```

INSERT INTO rental (
    id, staffid, customerid, inventoryid, rental_date)
VALUES (
    2, 1, 1, 4,
    to_timestamp('2023-12-11 06:14:00.742000000',
        'YYYY-MM-DD HH24:MI:SS.FF')
);
```

Script Output × | Task completed in 0.063 seconds

Error starting at line : 1 in command -

```

INSERT INTO rental (
    id, staffid, customerid, inventoryid, rental_date)
VALUES (
    2, 1, 1, 4,
    to_timestamp('2023-12-11 06:14:00.742000000',
        'YYYY-MM-DD HH24:MI:SS.FF')
)
```

Error report -

```

ORA-20500: You Can't Rental
the same Film Twice in a month
ORA-06512: at "HOMEWORKU.RENTAL_FILM_ONCE_TRIGGER", line 36
ORA-04088: error during execution of trigger 'HOMEWORKU.RENTAL_FILM_ONCE_TRIGGER'
```

نقوم بإضافة إيجار جديد من أجل العميل صاحب الـ ID رقم 1 للفلم الموجود لا `inventoryid` رقم 4 (وهو أيضاً الفلم الأول) فتفشل العملية لأنه قام بحجز الفلم أكثر من مرة في هذا الشهر:

الطلب الرابع: بناء سجل متابعة لعمليات الإضافة والتعديل لجدول الحجوزات في جدول رديف (يجب بناء الجدول) يسجل قيم جميع الحقول قبل وبعد تنفيذ العملية مع نوع وتاريخ و زمن العملية.

نقوم ببناء جدول المتابعة أولاً وتم اختصار حقول الـ `Foreign keys` الثلاثة في الجدول للسهولة فقط وإبقاء الحقول الثلاث الأخرى وثم نقوم ببناء أيضاً عدداً معرفة ترتيب عملية التتبع للجدول بالشكل التالي:

```
CREATE TABLE audit_rental (
    rental_id      NUMBER(10),
    old_rental_date TIMESTAMP,
    old_return_date TIMESTAMP,
    old_last_update TIMESTAMP,
    new_rental_date TIMESTAMP,
    new_return_date TIMESTAMP,
    new_last_update TIMESTAMP,
    audit_seq NUMBER(6),
    audit_kind VARCHAR2(1),
    audit_datetime TIMESTAMP
);
```

الآن نقوم ببناء القادح بالشكل التالي:

```
CREATE OR REPLACE TRIGGER audit_rental_values_trigger AFTER
    INSERT OR UPDATE ON rental
    FOR EACH ROW
DECLARE
    rental_id      NUMBER(10);
    v_audit_kind  VARCHAR2(1);
    v_audit_seq   NUMBER(6);
BEGIN
    IF inserting THEN
        v_audit_kind := 'I';
        rental_id :=:NEW.id;
    ELSIF updating THEN
        v_audit_kind := 'U';
        rental_id :=:OLD.id;
    END IF;
    SELECT auditseq.NEXTVAL INTO v_audit_seq FROM dual;
    INSERT INTO audit_rental VALUES (
        rental_id,
        :old.rental_date,
        :old.return_date,
        :old.last_update,
        :new.rental_date,
        :new.return_date,
        :new.last_update,
        v_audit_seq, v_audit_kind, sysdate);
END;
```

نقوم بإضافة الان سجل جديد الى جدول الحجوزات ومن ثم سنستعلم عن جدول التتبع الخاص بجدول الحجوزات

ونرى النتيجة:

```
= INSERT INTO rental (
    id, staffid, customerid,
    inventoryid, rental_date)
VALUES (2,1,1,2,sysdate
);
```

Script Output x  
Task completed in 0.034 seconds

1 row inserted.

`SELECT * FROM audit_rental;`

Query Result x

SQL | All Rows Fetched: 1 in 0.004 seconds

	RENTAL_ID	OLD_RENTAL_DATE	OLD_RETURN_DATE	OLD_LAST_UPDATE	NEW_RENTAL_DATE	NEW_RETURN_DATE	NEW_LAST_UPDATE	AUDIT_SEQ	AUDIT_KIND	AUDIT_DATETIME
1	2(null)	(null)	(null)	07-DEC-23...	(null)	(null)	(null)	1I	07-DEC-23 09.39.21.00000	

سنقوم بتعديل الحقل ونرى النتائج ايضاً:

```
UPDATE rental
SET
    return_date = sysdate
WHERE
    id = 2;|
```

Script Output x  
Task completed in 0.033 seconds

1 row updated.

`SELECT * FROM audit_rental;`

Query Result x

SQL | All Rows Fetched: 2 in 0.002 seconds

	RENTAL_ID	OLD_RENTAL_DATE	OLD_RETURN_DATE	OLD_LAST_UPDATE	NEW_RENTAL_DATE	NEW_RETURN_DATE	NEW_LAST_UPDATE	AUDIT_SEQ	AUDIT_KIND	AUDIT_DATETIME
1	2(null)	(null)	(null)	07-DEC-23...	(null)	(null)	(null)	1I	07-DEC-23 09.39.21.00000	
2	207-DEC-2...	(null)	(null)	07-DEC-23...	07-DEC-23...	(null)	(null)	2U	07-DEC-23 09.41.16.00000	

**الطلب الخامس: تسجيل زمن دخول وخروج أي مستخدم لقاعدة المعلومات في جدول مناسب ضمن حساب homeworku يشتري من ذلك حساب Sys**

```
GRANT ADMINISTER DATABASE TRIGGER
TO homeworku;
```

Grant succeeded.

Script Output X | Task completed in 0.032 seconds

نقوم اولاً بإعطاء حساب homeworku سماحية ADMINISTER DATABASE TRIGGER من حساب Sys ليتمكن المستخدم homeworku من إنشاء System Trigger قادر من نوع

نعود لحساب homeworku ونقوم بإنشاء جدول لتتبع عمليات تسجيل الدخول والخروج كالتالي:

```
CREATE TABLE system_log (
    operation VARCHAR2(30),
    who        VARCHAR2(30),
    when_date TIMESTAMP);
```

ثم نقوم بإنشاء قادحين الأول لا LOGON والأخر لا LOGOFF ولا يمكن جمعهم بقادح واحد لأن الا Logon لا تعمل إلا BEFORE Logoff ولا AFTER لا يعمل الا بالـ

```
CREATE OR REPLACE TRIGGER
track_logon_event_trigger AFTER LOGON ON
DATABASE BEGIN
    IF user NOT IN ('SYS', 'SYSTEM') THEN
        INSERT INTO system_log (operation, who,
        when_date) VALUES (ora_sysevent, user,
        sysdate);
    END IF;
END;
```

```
CREATE OR REPLACE TRIGGER
track_logoff_event_trigger BEFORE LOGOFF ON
DATABASE BEGIN
    IF user NOT IN ('SYS', 'SYSTEM') THEN
        INSERT INTO system_log (operation, who,
        when_date) VALUES (ora_sysevent, user,
        sysdate);
    END IF;
END;
```

select \* from system\_log;

Query... X

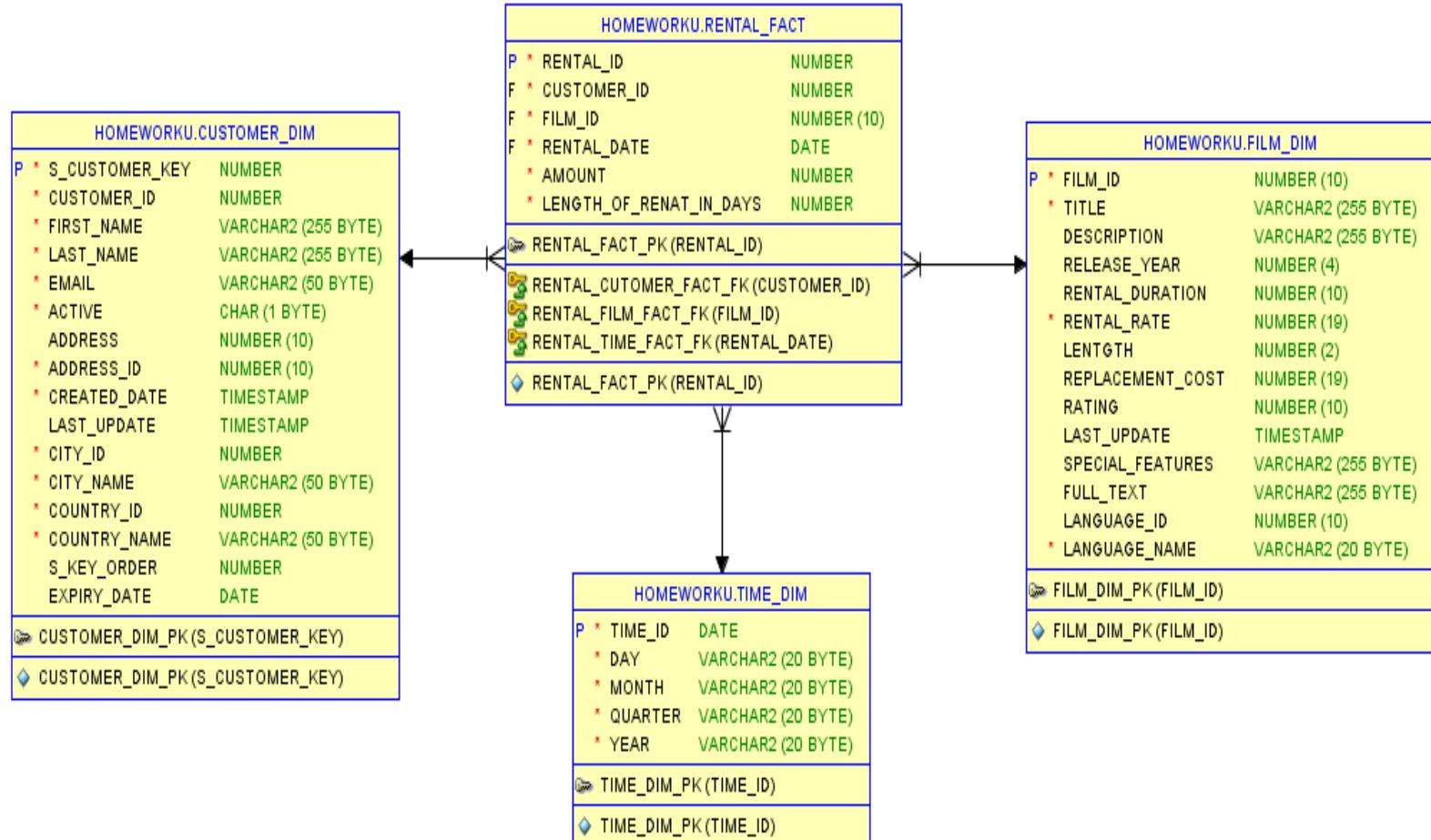
All Rows Fetched: 10 in 0.01 seconds

OPERATION	WHO	WHEN_DATE
1 LOGON	SCOTT	07-DEC-23 11.01.03.000000000 AM
2 LOGOFF	SCOTT	07-DEC-23 11.01.03.000000000 AM
3 LOGON	SCOTT	07-DEC-23 11.01.04.000000000 AM
4 LOGOFF	SCOTT	07-DEC-23 11.01.04.000000000 AM
5 LOGON	SCOTT	07-DEC-23 11.01.04.000000000 AM
6 LOGOFF	SCOTT	07-DEC-23 11.01.12.000000000 AM
7 LOGON	HR	07-DEC-23 10.58.05.000000000 AM
8 LOGOFF	HR	07-DEC-23 11.01.10.000000000 AM
9 LOGOFF	HOMWORKU	07-DEC-23 10.55.39.000000000 AM
10 LOGON	HOMWORKU	07-DEC-23 10.55.41.000000000 AM

الآن لو قمنا بتجريب تسجيل الدخول والخروج بعدة حسابات ومنها الا SYS AND SYSTEM وقمنا بالاستعلام التالي فستظهر النتائج كالتالي:

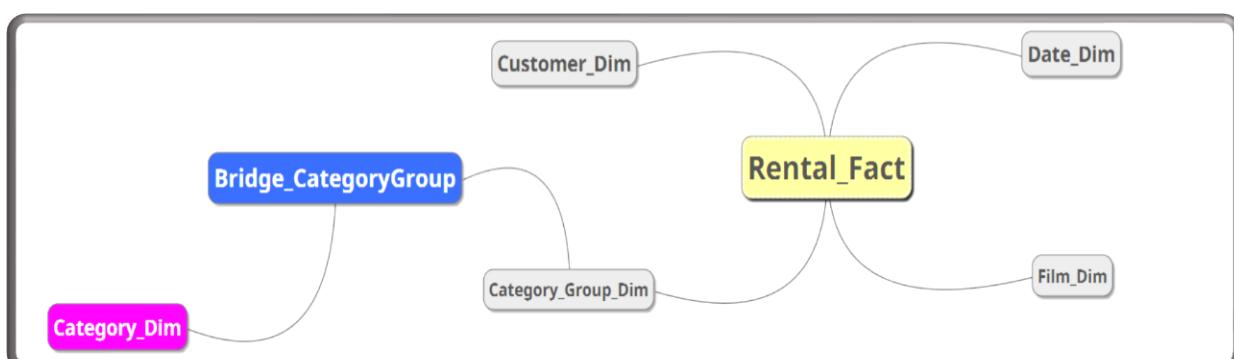
### القسم الثالث: قواعد المعطيات متعددة الابعاد

سنقوم ببناء المخطط من نوع **Star Schema** والذي سيمكنا من الحصول على اعلى اداء في الاستعلامات عن الاحصائيات ومحور الـ **Fact table** هما جدولتا **Payment And Rental** والمخطط بالشكل التالي:



#### ملاحظة مهمة لسبب اختيار المخطط والابعاد:

لم نقم بتضمين ابعاد نحن لسنا بحاجة لها من قراءتنا لنص الاسئلة في هذا القسم مثل الـ Staff, Store, etc. ومثلا ان كنا بحاجة لمعرفة فئات الأفلام Category وهي تملك علاقة Many-to-Many مع الـ Film هنا سنضطر لجعل المخطط Hybrid أي إضافة تفاصيل للأبعاد لن يتم البحث ضمنها الا عند الحاجة لتفاصيل فئات الفلم مثلا ونقوم على سبيل المثال باستخدام طريقة الـ Bridge Table كإحدى الطرق لحل هذه المشكلة والشكل التالي هو مجرد شكل لتوضيحي لكيف سيكون الشكل لمخطط:



### نقوم بإنشاء الجداول:

#### Film\_Dim

```
CREATE TABLE film_dim (
    film_id          NUMBER NOT NULL,
    title            VARCHAR2(255) NOT NULL,
    description      VARCHAR2(255),
    release_year     NUMBER(4),
    rental_duration  NUMBER(10),
    rental_rate      NUMBER(19, 0) NOT NULL,
    length           NUMBER(2),
    replacement_cost NUMBER(19, 0),
    rating           NUMBER(10),
    last_update      TIMESTAMP,
    special_features VARCHAR2(255),
    full_text        VARCHAR2(255),
    language_id      NUMBER(10),
    language_name    VARCHAR2(20) NOT NULL,
    CONSTRAINT film_dim_pk PRIMARY KEY (film_id)
);
```

#### Customer\_Dim

```
CREATE TABLE customer_dim (
    s_customer_key   NUMBER NOT NULL,
    customer_id      NUMBER NOT NULL,
    first_name       VARCHAR2(255) NOT NULL,
    last_name        VARCHAR2(255) NOT NULL,
    email            VARCHAR2(50) NOT NULL,
    active           CHAR(1) NOT NULL,
    created_date     TIMESTAMP NOT NULL,
    last_update      TIMESTAMP,
    address_id       NUMBER(10) NOT NULL,
    address          VARCHAR2(50) NOT NULL,
    city_id          NUMBER(10) NOT NULL,
    city_name        VARCHAR2(50) NOT NULL,
    country_id       NUMBER(10) NOT NULL,
    country_name     VARCHAR2(50) NOT NULL,
    s_key_order      NUMBER,
    expiry_date      DATE,
    CONSTRAINT customer_dim_pk PRIMARY KEY
        ( s_customer_key )
);
```

#### Time\_Dim

```
CREATE TABLE time_dim (
    time_id          DATE NOT NULL,
    day               VARCHAR2(20) NOT NULL,
    month             VARCHAR2(20) NOT NULL,
    quarter          VARCHAR2(20) NOT NULL,
    year              VARCHAR2(20) NOT NULL,
    CONSTRAINT time_dim_pk PRIMARY KEY (time_id)
);
```

#### Rental\_Fact

```
CREATE TABLE rental_fact (
    rental_id         NUMBER NOT NULL,
    customer_id       NUMBER NOT NULL,
    film_id           NUMBER NOT NULL,
    rental_date       DATE NOT NULL,
    amount             NUMBER NOT NULL,
    length_of_rental_in_days NUMBER NOT NULL,
    CONSTRAINT rental_fact_pk PRIMARY KEY (
        rental_id ),
    CONSTRAINT rental_customer_fact_fk FOREIGN KEY
        ( customer_id ) REFERENCES customer_dim
        ( s_customer_key ),
    CONSTRAINT rental_film_fact_fk FOREIGN KEY
        ( film_id ) REFERENCES film_dim ( film_id ),
    CONSTRAINT rental_time_fact_fk FOREIGN KEY
        ( rental_date ) REFERENCES time_dim ( time_id )
);
```

**الطلب الأول: بناء كافة الأبعاد مع مراعاة عملية التعديل التاريخية لبعد الزيون، وتحديد جميع الهرميات المناسبة لكل بعد.**

قمنا بإضافة Surrogate Key لجدول الزيون بالإضافة لحقلين إضافيين هما s\_key\_order و expiry\_date وبهذه الحقول سنقوم بحفظ جميع التعديلات التي ستم للزيون. سيتم ضافة حقل جديد عند أي تغيير في بيانات الزيون وستصبح قيم كل من s\_key\_order بقيمة ترتيب التعديل الذي طرأ أي ان كانت المرة الأولى فسيصبح 1 وللمرة الثانية 2 وهكذا وبالنسبة لـ expiry\_date ستتصبح بتاريخ حدوث التعديل وهذا الحال سيكونان بقيمة null لأن نسخة لبيانات الزيون وبالنسبة لـ key يوجد عدة طرق لتعبيته وهو ليس مرتبط بأي قيمة من الجدول الأساسي يمكن توليد قيمة من عدد Sequence ومالاحظة بهذه الطريقة المستخدمة سيترتب عبء علينا بالاستعلامات التي تخص الزيون لأن الزيون قد يوجد له أكثر من حقل في بعد الزيون.

### بناء الأبعاد وتحديد الهرميات:

#### Film\_Dim

#### Customer\_Dim

```
CREATE DIMENSION film_dim
  LEVEL film_id IS film_dim.film_id
  LEVEL title IS film_dim.title
  LEVEL description IS film_dim.description
  LEVEL release_year IS
    film_dim.release_year
  LEVEL rental_duration IS
    film_dim.rental_duration
  LEVEL rental_rate IS film_dim.rental_rate
  LEVEL length IS film_dim.length
  LEVEL replacement_cost IS
    film_dim.replacement_cost
  LEVEL rating IS film_dim.rating
  LEVEL last_update IS film_dim.last_update
  LEVEL special_features IS
    film_dim.special_features
  LEVEL full_text IS film_dim.full_text
  LEVEL language_id IS film_dim.language_id
  LEVEL language_name IS
    film_dim.language_name
  HIERARCHY hierarchy_film ( film_id
    CHILD OF language_id
  )
  ATTRIBUTE language_id DETERMINES (
    film_dim.language_name
);

```

```
CREATE DIMENSION customer_dim
  LEVEL s_customer_key IS
    customer_dim.s_customer_key
  LEVEL customer_id IS customer_dim.customer_id
  LEVEL first_name IS customer_dim.first_name
  LEVEL last_name IS customer_dim.last_name
  LEVEL email IS customer_dim.email
  LEVEL active IS customer_dim.active
  LEVEL created_date IS customer_dim.created_date
  LEVEL last_update IS customer_dim.last_update
  LEVEL address_id IS customer_dim.address_id
  LEVEL address IS customer_dim.address
  LEVEL city_id IS customer_dim.city_id
  LEVEL city_name IS customer_dim.city_name
  LEVEL country_id IS customer_dim.country_id
  LEVEL country_name IS
    customer_dim.country_name
  LEVEL s_key_order IS customer_dim.s_key_order
  LEVEL expiry_date IS customer_dim.expiry_date
  HIERARCHY hierarchy_customer (s_customer_key
    CHILD OF address_id CHILD OF city_id CHILD
    OF country_id)
    ATTRIBUTE address_id DETERMINES
      (customer_dim.address)
    ATTRIBUTE city_id DETERMINES
      (customer_dim.city_name)
    ATTRIBUTE country_id DETERMINES
      (customer_dim.country_name);
```

## Time\_Dim

```

CREATE DIMENSION time_dim
  LEVEL time_id IS time_dim.time_id
  LEVEL day IS time_dim.day
  LEVEL month IS time_dim.month
  LEVEL quarter IS time_dim.quarter
  LEVEL year IS time_dim.year
HIERARCHY hierarchy_time_all (day CHILD OF month CHILD OF quarter CHILD OF year)
HIERARCHY hierarchy_time (day CHILD OF month CHILD OF year);

```

نقوم بالتحقق من وجود الابعاد بـ التعليمـة التالية:

The screenshot shows the SQL Developer interface with a query window containing the following SQL statement:

```
SELECT * FROM all_dimensions;
```

The results pane displays the following table:

OWNER	DIMENSION_NAME	INVALID	COMPILE_STATE	REVISION
1 HOMEWORKU	FILM_DIM	N	VALID	1
2 HOMEWORKU	TIME_DIM	N	VALID	1
3 HOMEWORKU	CUSTOMER_DIM	N	VALID	1

**الطلب الثاني:** تجزئة جدول Fact بالطريقة التي تراها مناسبة وتحليل ذلك، هل من الممكن أن تكون هذه التجزئة مركبة وضح ذلك.

نقوم بعمل Drop لجدول Fact بحالتنا لأنه خالي من أية سجلات ثم نقوم ببنائه من جديد وبـ Partition معين لكن ان كان يحوي سجلات فيجب عمل تصدير لها ثم Drop للجدول وانشاءه بالتقسيمة المراده من جديد وعمل استيراد للبيانات وذلك لعدم القدرة على انشاء اقسام (Partitions) للجدول بعد انشائه (وذلك لجميع نسخ oracle التي دون (12c).

نستخدم **PARTITION BY RANGE** على جدول Fact وذلك للحقل rental\_date كل سنة لحال ويمكن التقسيم كل شهر او او .. الخ ولكن سنعتمد على كل سنة بقسم لحال نظرا الى نوع البيانات لدينا فهي حجوزات أفلام فمن المنطق انه مهما بلغ عدد الحجوزات كبير لن تصل الى ارقام ضمن السنة الواحدة ستبطئ عمل الاستعلامات

بالشكل التالي:

```

CREATE TABLE rental_fact (
    rental_id      NUMBER NOT NULL,
    customer_id    NUMBER NOT NULL,

```

```

film_id      NUMBER NOT NULL,
rental_date   DATE NOT NULL,
amount        NUMBER NOT NULL,
length_of_rental_in_days NUMBER NOT NULL,
CONSTRAINT rental_fact_pk PRIMARY KEY ( rental_id ),
CONSTRAINT rental_customer_fact_fk FOREIGN KEY ( customer_id )
REFERENCES customer_dim (s_customer_key ),
CONSTRAINT rental_film_fact_fk FOREIGN KEY ( film_id ) REFERENCES
film_dim (film_id ),
CONSTRAINT rental_time_fact_fk FOREIGN KEY ( rental_date )
REFERENCES time_dim (time_id )
)
PARTITION BY RANGE (rental_date)
(
    PARTITION rentals_2020 VALUES LESS
        THAN(TO_DATE('01/01/2020','DD/MM/YYYY')),
    PARTITION rentals_2021 VALUES LESS
        THAN(TO_DATE('01/01/2021','DD/MM/YYYY')),
    PARTITION rentals_2022 VALUES LESS
        THAN(TO_DATE('01/01/2022','DD/MM/YYYY')),
    PARTITION rentals_2023 VALUES LESS
        THAN(TO_DATE('01/01/2023','DD/MM/YYYY')),
    PARTITION rentals_after_2023 VALUES LESS THAN(MAXVALUE)
);

```

يمكن عمل تجزئة مركبة بمثابة عن طريق **RANGE - HASH** حيث من الممكن ان نرى ان تجزئة كل سنة قد تحوي الكثير من البيانات فيمكن إضافة `hash` وسيكون لحقل `rental_id` ولكن التقسيمات للجدول ترى من حجم العمل والتوقعات للBusiness.

سنستعرض الأقسام للجدول التي تم انشائها:

PARTITION_NAME	LAST_ANALYZED	NUM_ROWS	BLOCKS	SAMPLE_SIZE	HIGH_VALUE
1 RENTALS_2020	(null)	(null)	(null)	(null)	TO_DATE(' 2020-01-01 00:00:00 ',
2 RENTALS_2021	(null)	(null)	(null)	(null)	TO_DATE(' 2021-01-01 00:00:00 ',
3 RENTALS_2022	(null)	(null)	(null)	(null)	TO_DATE(' 2022-01-01 00:00:00 ',
4 RENTALS_2023	(null)	(null)	(null)	(null)	TO_DATE(' 2023-01-01 00:00:00 ',
5 RENTALS AFTER 2023	(null)	(null)	(null)	(null)	MAXVALUE

**الطلب الثالث:** بناء MATERIALIZED VIEW يتضمن اسم الفيلم واسم الزبون وفترة الحجز، تتحدد بيانات هذا المنظور بشكل تراكمي.

نقوم بإنشاء MV LOG أولاً لكل الجداول التي سنأخذ منها البيانات وهي customer\_dim, film\_dim and rental\_fact وهي بالشكل التالي:

```
CREATE MATERIALIZED VIEW LOG ON customer_dim WITH
  PRIMARY KEY,
  ROWID
  INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON film_dim WITH
  PRIMARY KEY,
  ROWID
  INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON rental_fact WITH
  PRIMARY KEY,
  ROWID
  INCLUDING NEW VALUES;
```

وسبب حاجتنا لهذه LOG VIEW بسبب ماتم طلبه بالسؤال من التحديث بشكل تراكمي فكل فترة سنقوم بعمل refresh ليتم تحديث الـ MATERIALIZED VIEW التي سننشئها تالياً:

```
CREATE MATERIALIZED VIEW film_rental_view
  BUILD IMMEDIATE REFRESH FAST ON DEMAND
AS
  SELECT
    f.rowid film_rid, c.rowid custoemr_rid, r.rowid rental_rid,
    f.title          film_name,
    ( c.first_name || ' ' || c.last_name ) customer_name,
    r.length_of_renat_in_days
  FROM
    film_dim f, customer_dim c, rental_fact r
  WHERE
    r.customer_id = c.s_customer_key
    AND r.film_id = f.film_id;
```

وضعنا الـ rowid لأنها اجبارية اذا كنا سنقوم بعمل MV من نوع REFRESH FAST والـ MV لا يحوي الـ JOINs بدون آية توابع تجميعية (<https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/basic-materialized-views.html%23GUID-8823A06E-853B-4876-AB9F-96D2D4E5A1DE>)

نستعرض الـ MV بالشكل التالي:

```
select film_name, customer_name, length_of_renat_in_days from film_rental_view;
```

Script Output x | Query Result x  
SQL | Fetched 50 rows in 0.009 seconds

FILM_NAME	CUSTOMER_NAME	LENGTH_OF_RENAT_IN_DAYS
1 Movie 10	First Name 32 Last Name 32	32
2 Movie 16	First Name 85 Last Name 85	21
3 Movie 9	First Name 73 Last Name 73	4
4 Movie 3	First Name 82 Last Name 82	43
5 Movie 7	First Name 38 Last Name 38	24
6 Movie 6	First Name 80 Last Name 80	38
7 Movie 11	First Name 22 Last Name 22	14
8 Movie 7	First Name 64 Last Name 64	30
9 Movie 17	First Name 38 Last Name 38	38
0 Movie 5	First Name 91 Last Name 91	45
1 Movie 10	First Name 69 Last Name 69	22
2 Movie 5	First Name 43 Last Name 43	12
3 Movie 9	First Name 77 Last Name 77	41

الطلب الرابع: عرض قائمة بقيم حجوزات الأفلام على مستوى الشهر والبلد والمدينة بكافة الاحتمالات الممكنة.

نقوم بكتابة الاستعلام بالشكل التالي وعرض النتيجة:

SELECT

```

decode(GROUPING(f.title), 1, 'Multi-Films SUM', f.title) AS film_title,
decode(GROUPING(t.month), 1, 'Multi-Months SUM', t.month) AS month,
decode(GROUPING(c.country_name), 1, 'Multi-Countries SUM', c.country_name) AS country,
decode(GROUPING(c.city_name), 1, 'Multi-Cities SUM', c.city_name) AS city,
to_char(SUM(r.amount),'9,999,999,999') AS total_amount
FROM
    rental_fact r, film_dim f, customer_dim c, time_dim t
WHERE
    r.customer_id = c.s_customer_key AND r.film_id = f.film_id AND r.rental_date = t.time_id
GROUP BY CUBE(f.title, t.month, c.country_name, c.city_name)
ORDER BY t.month, c.country_name, c.city_name, f.title;

```

Query Result x

SQL | Fetched 50 rows in 0.07 seconds

FILM_TITLE	MONTH	COUNTRY	CITY	TOTAL_AMOUNT
1 Movie 11	April	Country 1	City 4	1,469
2 Movie 12	April	Country 1	City 4	4,225
3 Multi-Films SUM	April	Country 1	City 4	5,694
4 Movie 18	April	Country 1	City 54	1,006
5 Movie 4	April	Country 1	City 54	707
6 Movie 7	April	Country 1	City 54	2,551
7 Movie 9	April	Country 1	City 54	4,069
8 Multi-Films SUM	April	Country 1	City 54	8,333
9 Movie 11	April	Country 1	Multi-Cities SUM	1,469
10 Movie 12	April	Country 1	Multi-Cities SUM	4,225
11 Movie 18	April	Country 1	Multi-Cities SUM	1,006
Movie 4	April	Country 1	Multi-Cities SUM	707

الطلب الخامس: أوجد ترتيب كل سنة وكل فيلم ضمن هذه السنة حسب قيمة التأجير.

نقوم بكتابة الاستعلام بالشكل التالي وعرض النتيجة:

SELECT

```
to_char(r.RENTAL_DATE, 'yyyy') AS Year, f.title AS film,
SUM(r.amount) OVER (PARTITION BY to_char(r.RENTAL_DATE, 'yyyy')) AS Year_Total,
SUM(r.amount) OVER (PARTITION BY to_char(r.RENTAL_DATE, 'yyyy'),
                     to_char(r.RENTAL_DATE, 'mm')) AS Month_Total,
DENSE_RANK() OVER (PARTITION BY to_char(r.RENTAL_DATE, 'yyyy')
                    ORDER BY r.amount) AS Year_Rank,
DENSE_RANK() OVER (PARTITION BY to_char(r.RENTAL_DATE, 'yyyy'),
                    to_char(r.RENTAL_DATE, 'mm') ORDER BY r.amount) AS Month_Rank
FROM rental_fact r, film_dim f WHERE r.film_id = f.film_id;
```

cript Output x | Query Result x

SQL | Fetched 150 rows in 0.022 seconds

	YEAR	FILM	YEAR_TOTAL	MONTH_TOTAL	YEAR_RANK	MONTH_RANK
10	2020	Movie 10	3421225	278598	8	2
11	2020	Movie 1	3421225	300503	9	1
12	2020	Movie 12	3421225	352504	10	1
13	2020	Movie 9	3421225	285304	11	1
14	2020	Movie 7	3421225	278598	12	3
15	2020	Movie 8	3421225	284834	13	2
16	2020	Movie 18	3421225	246006	14	1
17	2020	Movie 19	3421225	278598	15	4
18	2020	Movie 4	3421225	352504	16	2
19	2020	Movie 9	3421225	285304	17	2
20	2020	Movie 5	3421225	352504	18	3
21	2020	Movie 11	3421225	270520	19	2
22	2020	Movie 15	3421225	278598	20	5
23	2020	Movie 17	3421225	270520	21	3
	2020	Movie 4	3421225	278598	22	6

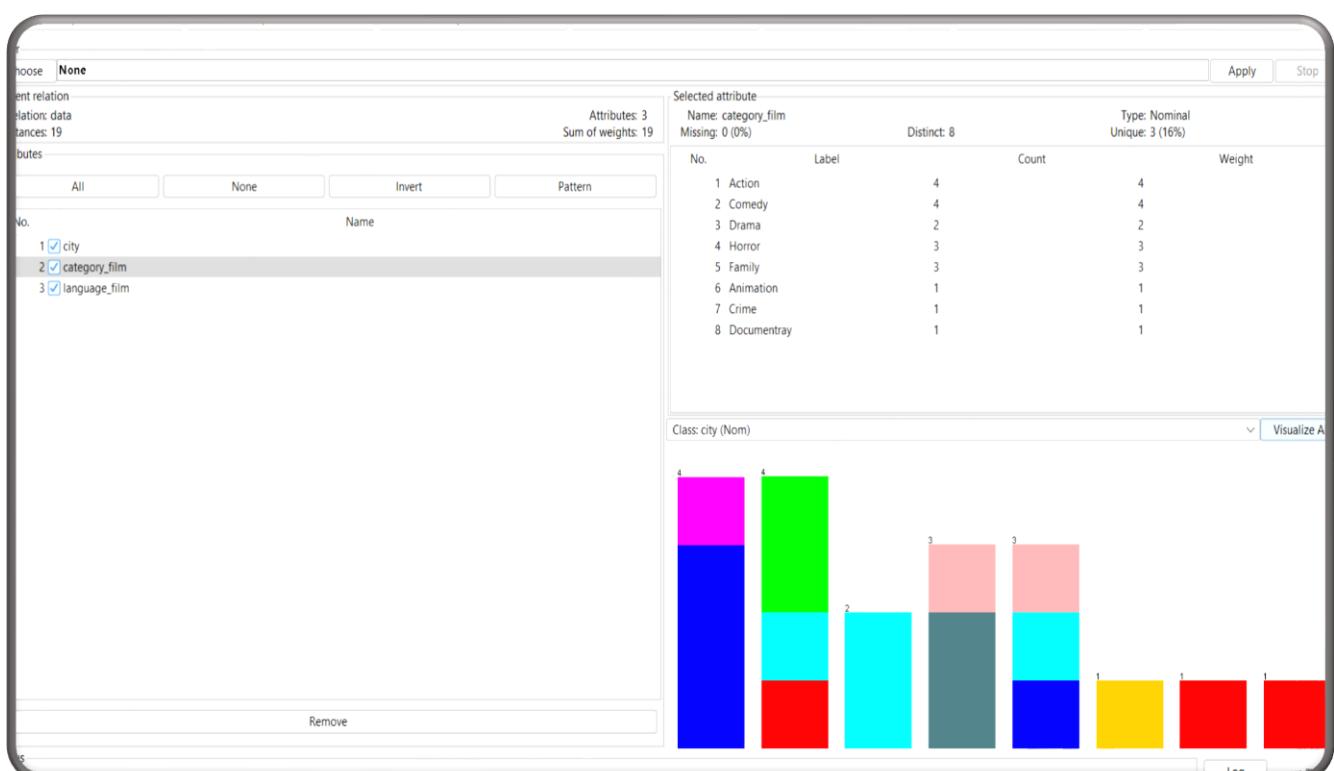
**الطلب السادس:** يريد مدير الشركة أن يعرف ما هي العلاقة ما بين مدينة الربون وفئة الأفلام ولغة الفيلم لكي يستخدم هذه المعلومة في الإعلانات الموجبة، اقترح طريقة مناسبة لمعرفة هذه العلاقة ثمنفذها باستخدام .weka

نملأ البيانات في ملف Excel ونحفظه بصيغة CSV ثم نقوم بفتحه باستخدام weka ونقوم بتحويل الملف من صيغة ARFF إلى صيغة CSV

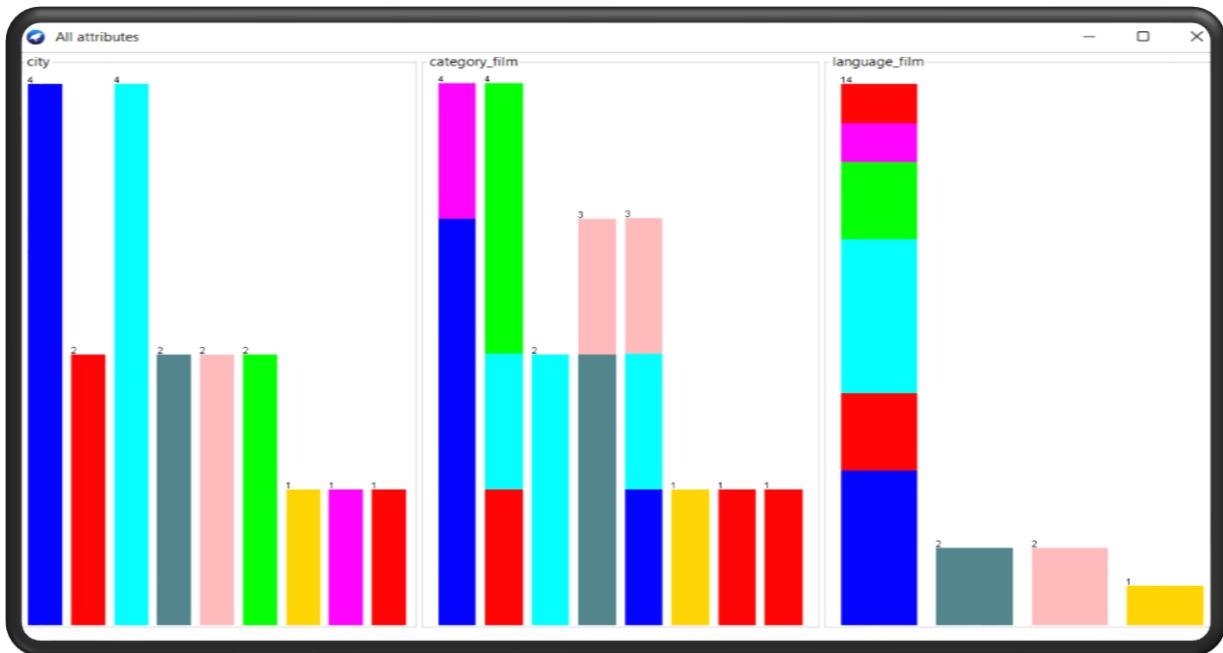
C	B	A
language_film	category_film	city
English	Action	Cario
English	Comedy	New York City
English	Drama	London
French	Horror	Paris
Italian	Family	Rome
English	Comedy	Sydney
English	Action	Cario
Italian	Horror	Rome
Russian	Animation	Moscow
English	Crime	New York City
English	Comedy	London
English	Action	Dubai
English	Action	Cario
English	Drama	London
English	Documentray	Los Angeles
French	Horror	Paris
English	Family	Cario
English	Family	London
English	Comedy	Sydney

البيانات ضمن Excel

فيكون الخرج كالتالي:



وبتحليل جميع البيانات ينتج:



وبتنفيذ خوارزمية apriori على البيانات المدخلة تكون:

```

Associator output

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: data
Instances: 19
Attributes: 3
    city
    category_film
    language_film
==== Associator model (full training set) ====

Apriori
=====

Minimum support: 0.1 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 14

Size of set of large itemsets L(2): 15

Size of set of large itemsets L(3): 4

Best rules found:

1. city=Cario 4 ==> language_film=English 4   <conf:(1)> lift:(1.36) lev:(0.06) [1] conv:(1.05)
2. city=London 4 ==> language_film=English 4   <conf:(1)> lift:(1.36) lev:(0.06) [1] conv:(1.05)
3. category_film>Action 4 ==> language_film=English 4   <conf:(1)> lift:(1.36) lev:(0.06) [1] conv:(1.05)
4. category_film=Comedy 4 ==> language_film=English 4   <conf:(1)> lift:(1.36) lev:(0.06) [1] conv:(1.05)
5. city=Cario category_film=Action 3 ==> language_film=English 3   <conf:(1)> lift:(1.36) lev:(0.04) [0] conv:(0.75)
6. city>New York City 2 ==> language_film=English 2   <conf:(1)> lift:(1.36) lev:(0.03) [0] conv:(0.53)
7. category_film=Drama 2 ==> city=London 2   <conf:(1)> lift:(4.75) lev:(0.08) [1] conv:(1.58)
8. city=Paris 2 ==> category_film=Horror 2   <conf:(1)> lift:(6.33) lev:(0.09) [1] conv:(1.68)
9. language_film=French 2 ==> city=Paris 2   <conf:(1)> lift:(9.5) lev:(0.09) [1] conv:(1.79)
10. city=Paris 2 ==> language_film=French 2   <conf:(1)> lift:(9.5) lev:(0.09) [1] conv:(1.79)

```