# Security; Integrity

Week 07

# How to do give access?

GRANT privilege_name
ON object name
TO {user name | PUBLIC | role name}
[WITH GRANT OPTION];

Or GRANT a list of privileges, to a list of users, in one command!

# What privileges can we grant?

- SELECT
  - Allows someone to run select statements on the object
- INSERT
  - Allows someone to insert new rows into the object
- DELETE
  - Allows someone to delete existing rows in the object
- UPDATE
  - Allows someone to change values in existing rows in the object
- REFERENCES
  - Allows someone to define a foreign key constraint in a table they own, that references the key of the object
- EXECUTE
  - Allows someone to run the stored procedure if the object is a stored procedure
- ALL
  - Allows someone to do all of the above to the object

# Who can we give privileges to?

- User_name
  - A specific user
- role_name
  - The set of users who have been granted to belong to role_name
- PUBLIC
  - Everyone, usually goes by different names in different DBMSs
    - PUBLIC, everyone, all_users

# How can we give contagious privileges

- Add [WITH GRANT OPTION] to the end of your grant statement to allow the user to pass on any privileges they have been granted.
- Always be careful when granting ALL or UPDATE/DELETE WITH GRANT OPTION as you can soon lose track of who has what privileges (although the system keeps track and you can run a query to see who has them)

# Working in PostgreSQL

- As well as granting access on a table etc, you need to also grant USAGE access to the schema that contains the table eg GRANT USAGE ON unidb to Joe;
- When opening the connection, the user needs to be sure to connect to the database in which the table exists

# Schema

- Book (isbn, title, publisher, publicationYear)
- Author (aname, birthdate)
- Publisher (pname, address)
- Wrote (isbn, aname) // this book was (perhaps co-)written by this author

What command is needed, if we want to permit Albert to find out about Books?

GRANT SELECT

ON Book

TO Albert;

Book (isbn, title, publisher, publicationYear)
Author (aname, birthdate)
Publisher (pname, address)
Wrote (isbn, aname)

# permit Bob to update the database by recording information about additional Books

- What command is needed, if we want to permit Bob to update the database by recording information about additional Books, but he can't change or inspect existing information?
  - i. Further discussion: what might be a difficulty arising from this policy, for Bob (e.g. if he wants to check what he has just done?

Book (isbn, title, publisher, publicationYear)
Author (aname, birthdate)
Publisher (pname, address)
Wrote (isbn, aname)

GRANT INSERT ON Book TO Bob;

GRANT INSERT ON Wrote TO Bob;

Maybe these as well (otherwise Bob can't include a book with an author/publisher who was not previously known):

GRANT INSERT ON Author TO Bob;

GRANT INSERT ON Publisher TO Bob;

(i) Wise to also grant SELECT on Book so Bob can check that all inserts are correct

permit Chenyi to correct mistakes among the author birthdates?

Book (isbn, title, publisher, publicationYear)
Author (aname, birthdate)
Publisher (pname, address)
Wrote (isbn, aname)

```
GRANT SELECT,
UPDATE(birthdate)
ON Author
TO Chenyi;
```

GRANT SELECT so Chenyi can see the mistakes, to know what to fix

# permit Dora to see the information for books whose publisher is “WH Smith”

- [Hint: create a view]

```
CREATE VIEW WHSmithBooks AS
(SELECT title,isbn,publicationYear
FROM Book
WHERE publisher = ‘WH Smith’);
GRANT SELECT on WHSmithBooks TO Dora;
```

# permit Emily to see how many books each publisher plans for future

- [Hint: create a view]

```
CREATE VIEW FutureBooksByPub AS
(SELECT publisher, count(distinct isbn)
FROM Book
WHERE publicationyear > 2022
GROUP BY publisher);
GRANT SELECT on FutureBooksByPub TO Emily;
```

(a) Give some sample CREATE TABLE Statements to represent the schema out-lined in Exercise 1 with all the primary key and foreign keys represented as constraints.

```
CREATE TABLE Publisher(
    pname varchar(50),
    address varchar(250),
    CONSTRAINT PK_publisher PRIMARY KEY(pname)
);
```

```sql
CREATE TABLE Book(
    isbn char(13),
    title varchar(250),
    publisher varchar(50),
    publicationYear int,
    CONSTRAINT PK_book PRIMARY KEY(isbn),
    CONSTRAINT FK_pname_publisher FOREIGN KEY (publisher)
        REFERENCES Publisher(pname)
        -- ON DELETE no action
        -- ON UPDATE no action
);
```

```
CREATE TABLE Author(
    aname varchar(50),
    birthdate DATE,
    CONSTRAINT PK_Author PRIMARY KEY(aname)
);
-- have an assumption that no two authors have the same name
```

```
CREATE TABLE Wrote(
    isbn char(13),
    aname varchar(50),
    CONSTRAINT FK_isbn_book FOREIGN KEY (isbn)
        REFERENCES Book(isbn),
        -- ON DELETE no action
        -- ON UPDATE no action
    CONSTRAINT FK_aname_author FOREIGN KEY (aname)
        REFERENCES Author(aname)
        -- ON DELETE no action
        -- ON UPDATE no action
    CONSTRAINT PK_Author PRIMARY KEY(isbn,aname)
);
```

a). All books you may have in the system are published from year 1997 to2022,

```
ALTER TABLE Book
ADD CONSTRAINT yearslimit
CHECK ( ( publicationYear >= 1997)
AND ( publicationYear <= 2022));
-- maybe instead use CHECK ( publicationYear between 1997 and 2022);
--
```

b). any book with publisher BrowseBooks must have a publication year no later than 2015 ,

```
ALTER TABLE Book
ADD CONSTRAINT pubyearslimit
CHECK ( (publisher <> 'BrowseBooks') OR ( publicationYear <= 2015))
```

c). any book must have a publication year at least 10 years after the birthdate of any author,

```
ALTER TABLE Book
ADD CONSTRAINT nochildauthors
CHECK (  publicationYear >= (SELECT MAX(10+EXTRACT(YEAR FROM birthdate))
                             FROM Author A, Wrote W
                             WHERE A.authorname = W.authorname
                                   AND W.isbn = isbn)
```

-- Subquery in CHECK constraint is not supported by PostgreSQL