Views, Levels of Abstraction and Data Independence

One database often supports multiple applications, which might have slightly different pictures of the world.

Views help accommodate this variation without storing redundant data.

Views

- A view is a query with a name that is stored in the DB.

- You can use a view as if it were a table. You can use it in queries.

Example view definition:

```
CREATE VIEW GStudent AS
SELECT *
FROM Student S
WHERE s.gpa >= 2.5
```

Example view use: Simpler queries

- Completed(StudID, Course)

- Queries are often about “good” students only

```
SELECT S.name, S.phone
FROM GStudent S NATURAL JOIN Completed C
WHERE C.course = ‘CS386’;
```

- Now it’s easier to write the query.

Views for Security

Suppose schema is

Student(studID, name, address, major, gpa)

- This is a view of the Student table without the gpa field.

```
CREATE VIEW SecStudent AS
SELECT studID, name, address, major
FROM student
```

Views for Extensibility

- A company's database includes a relation: Part (PartID: Char(4), weight:real,…)
- Weight is stored in pounds
- Company is purchased by a firm that uses metric weights
- Databases must be integrated and use Kg.
- But there's much old software using pounds.
- Solution: views!

Views for extensibility (cont.)

- Solution:

1. Base table with kilograms becomes NewParts, for integrated company.
2. CREATE VIEW Part AS
   SELECT PartID, 2.2046*weight, …(*no other changes*)…
   FROM NewParts

3. Old programs still call the table “Part”

Quick Exercise

- Define a view in your postgresql DB. Use just one table and omit some attributes in the view.
- Issue one or more queries against the view that you have just defined.
- Define a view that uses join in its definition.
- Define a view that uses group by in its definition.
- Issue one or more queries against your views.
- Write a query that involves one of your views twice or that joins your view(s) with other tables.

Another Exercise

- For each of your views, discuss whether or not it is possible to insert (or update or delete) a row that appears in a view.
- Attempt to insert a new row into the last view that you defined above (the one that is essentially identical to the original/base table); see if postgresql allows you to do so – directly.

But there's one problem with views

- Views cannot always be updated unambiguously
- For this table: Student(StudID, gpa, major,…) define this view:

```
CREATE VIEW majorgpa AS
SELECT major, AVG(gpa)
FROM Student
GROUP BY major
```

```
select * from Majorgpa
```

major	gpa
CS	3.5
ECE	3.5

Updatability of views

- I want to change the GPA of CS majors from 3.5 to 3.6 .
- How can I do that?

A view can be updated if:

- It is defined on a single base table
- It includes all of the attributes that must have values
- It has no subqueries
- It has no aggregates
- It does not use DISTINCT

Quick Exercise:

Talk to your partner and discuss why an updateable view definition can't include "DISTINCT"

Talk to your partner and discuss why an updateable view definition can't include subqueries.

A few Comments on Views

- View versus temporary table
  - What is the difference?
  - Which is better?
- Some DBMS' offer a way around the view update problem
  - Triggers (particularly "instead of" triggers)

How are views implemented?

When you enter a query that mentions a view in the from clause, the DBMS expands/rewrites your query to include the view definition.

```
SELECT S.name, S.phone
FROM gstudent S NATURAL JOIN completed C
WHERE C.course = 'CS386';
```

is rewritten as:

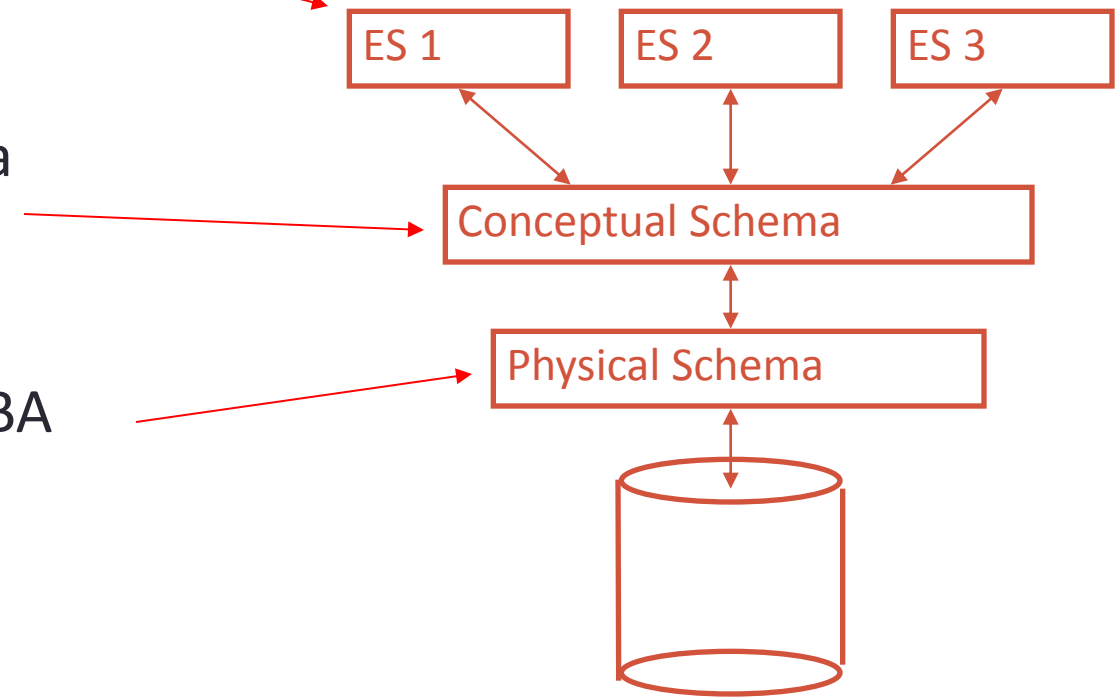
```
SELECT S.name, S.phone
FROM (SELECT S.* FROM student S  WHERE s.gpa >= 2.5) AS S
  NATURAL JOIN completed C
WHERE C.course = 'CS386';
```

Levels of Abstraction

External view; user and data designer

Logical storage; data designer

Physical storage; DBA

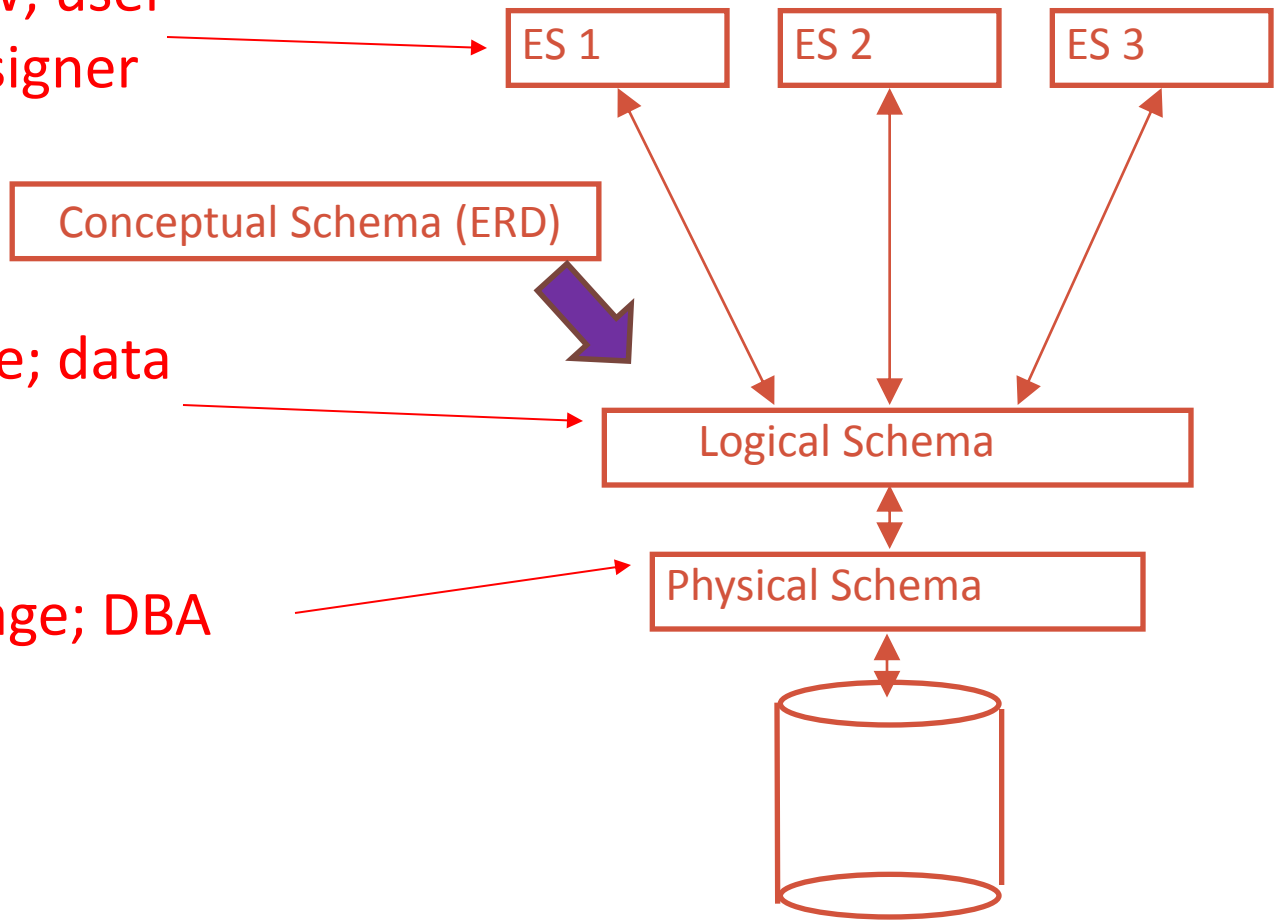


Levels of Abstraction (refined)

External view; user and data designer

Logical storage; data designer

Physical storage; DBA



Physical Schema

The *physical schema* is a description of how the data is physically stored in the database. It includes

- Where the data is located
- File structures
- Access methods
- Indexes

The physical schema is managed by the DBA.

Logical Schema

The logical schema is a description of what data is in the database.

It consists of the schemas we have described with CREATE TABLE statements.

External Schemas

Each external schema is a combination of base tables and views, tailored to the needs of a single user. It is managed by the data designer and the user.

Data Independence

- A database model exhibits *data independence* if: application programs are protected from changes in the conceptual and physical schemas.
- Why is this important? Everything changes.
- How does the relational model achieve logical (conceptual) data independence?

Data Independence (ctd.)

- How does the relational model achieve physical data independence?
  1. Conceptual level contains no physical info
  2. SQL can program against the logical level
- Earlier DBMSs (network, hierarchical) did not have these properties.
  - Their languages had physical properties embedded in them.

Summary

- A view is a stored query definition
- Views can be very useful
  - Easier query writing, security, extensibility
- But not all views can be updated unambiguously
- Three levels of abstraction in a relational DBMS
  - Yields data independence: logical and physical