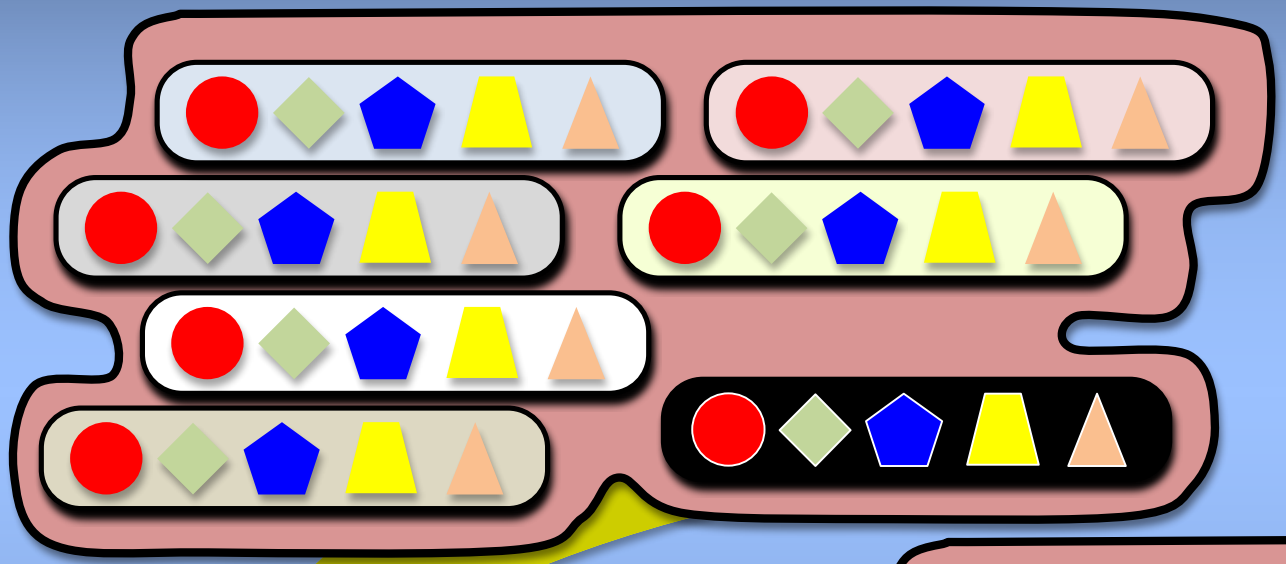
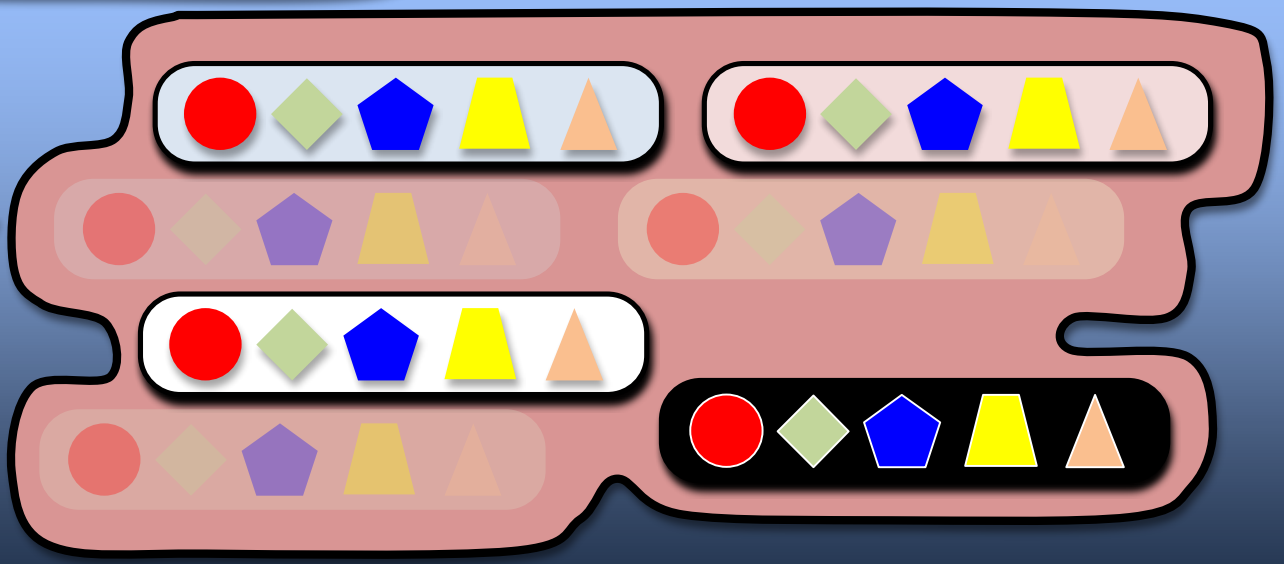

Tony Veale, UCD
Database Views in SQL

# Relational Algebra Constructs "Viewpoints"

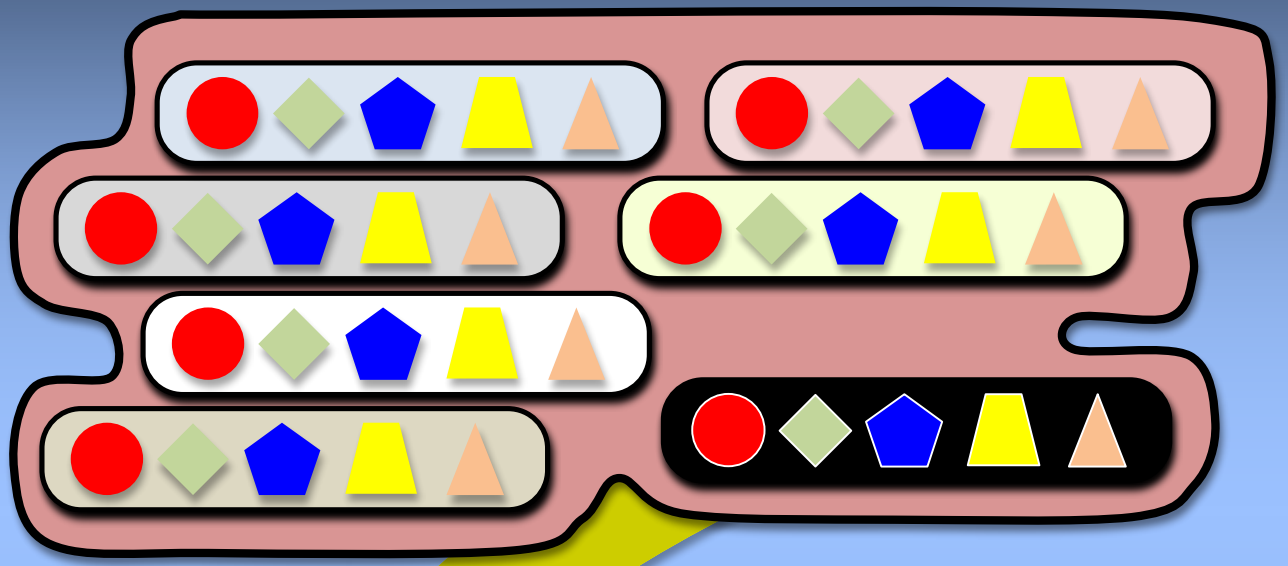


A **view** is a highly selective perspective on a database table (or tables)

Relational algebraic form

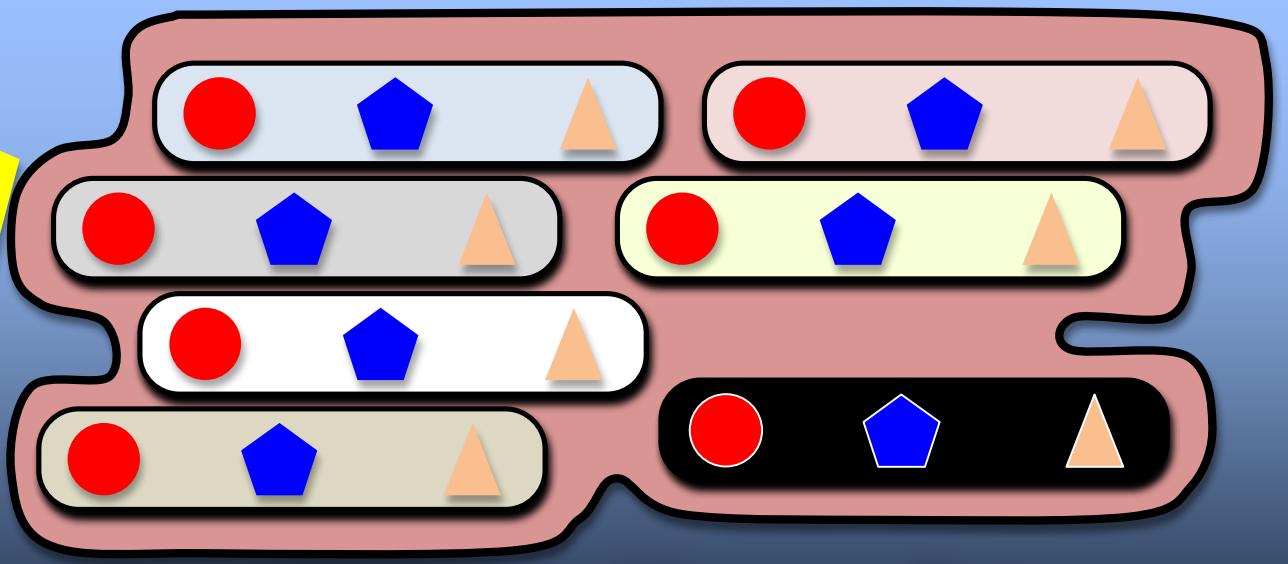


# What If We Treat SQL Queries as Dynamic Views?



A result set is a *static* view. An SQL query offers a *dynamic* view.

View-creating SQL query



Alternate View on Data

# A "View" Creates Perspectives On Existing Tables

## Entities (base table)

| Name | Gender | Fictive | Opponent |
|---|---|---|---|
| **Adam Smith** | *male* | *no* | *Karl Marx* |
| **Princess Leia** | *female* | *yes* | *Darth Vader* |
| **Hillary Clinton** | *female* | *no* | *Donald Trump* |
| **Lex Luthor** | *male* | *yes* | *Superman* |

# Let's Create TWO Views on this Table

## Fictional Entities & Real Entities ...

# Two Queries Allow Us to Populate Two Tables

## Real_Entities

| Name | Gender | Opponent |
|---|---|---|
| **Adam Smith** | *male* | *Karl Marx* |
| **Hillary Clinton** | *female* | *Donald Trump* |

## Fictional_Entities

| Name | Gender | Opponent |
|---|---|---|
| **Princess Leia** | *female* | *Darth Vader* |
| **Lex Luthor** | *male* | *Superman* |

# But These Tables Do Not Grow as Their Basis Grows

# A View Is a Query That Thinks It's a Table

```
CREATE VIEW view_name AS
SELECT column1, column2, ....
FROM table_name
WHERE {condition};
```

```
CREATE VIEW Real_Entities AS
SELECT Name, Gender, Opponent
FROM Entities
WHERE Fictive = "no";
```

```
CREATE VIEW Fictional_Entities AS
SELECT Name, Gender, Opponent
FROM Entities
WHERE Fictive = "yes";
```

```
SELECT Name, Opponent
FROM Real_Entities
WHERE Gender = "male";
```

```
SELECT Name, Opponent
FROM Fictional_Entities
WHERE Gender = "female";
```

| Name | Opponent |
|---|---|
| Adam Smith | *Karl Marx* |
| Bill Clinton | *Newt Gingrich* |
| Donald Trump | *Hillary Clinton* |

| Name | Opponent |
|---|---|
| Princess Leia | *Emperor Palpatine* |
| Lara Croft | *Lord HawHaw* |
| Jessica Jones | *Purple Man* |

# Views Can Be Queried Just Like Tables

```
UPDATE Real_Entities
SET Opponent = "Paul Ryan"
WHERE Name = "Donald Trump";
```

## Entities (base table)

| Name | Gender | Fictive | Opponent |
|---|---|---|---|
| **Adam Smith** | *male* | *no* | *Karl Marx* |
| **Princess Leia** | *female* | *yes* | *Darth Vader* |
| **Donald Trump** | *female* | *no* | ~~*Hillary*~~ ***Paul Ryan*** |
| **Lex Luthor** | *male* | *yes* | *Superman* |

# Updates to the View Are Made to the Base Table

```
DELETE FROM Fictional_Entities
WHERE Name = "James Bond";
```

## Entities (base table)

| Name | Gender | Fictive | Opponent |
|---|---|---|---|
| **Adam Smith** | *male* | *no* | *Karl Marx* |
| **Princess Leia** | *female* | *yes* | *Darth Vader* |
| **Donald Trump** | *female* | *no* | *Paul Ryan* |
| **Lex Luthor** | *male* | *yes* | *Superman* |
| ~~**James Bond**~~ | ~~*male*~~ | ~~*yes*~~ | ~~*Ernst Stavro Blofeld*~~ |

# Deletion from a View entails Deletion from the Base

```
INSERT INTO Fictional_Entities
VALUES ("Jason Bourne", "male",
        "Carlos the Jackal");
```

| Name | Gender | Fictive | Opponent |
|---|---|---|---|
| **Adam Smith** | *male* | *no* | *Karl Marx* |
| **Princess Leia** | *female* | *yes* | *Darth Vader* |
| **Hillary Clinton** | *female* | *no* | *Donald Trump* |
| **Lex Luthor** | *male* | *yes* | *Superman* |
| **Jason Bourne** | *male* | ***NULL*** | *Carlos the Jackal* |

# Columns "Outside" the View Must Accept Null Values

# Views may be based on arbitrarily complex queries ...
# ... but only SIMPLE Views can be Modified!

- The SELECT clause may not use the keyword DISTINCT.
- The SELECT clause may not use *aggregate* functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not have an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain sub-queries.
- The query may not contain GROUP BY or HAVING.
- *Calculated* columns may not be updated.
- All NOT NULL columns from the base table must be included in the view for the INSERT query to function.

```
CREATE VIEW Fictional_Entities AS
SELECT Name, Gender, Opponent
FROM Entities
WHERE Fictive = "yes"
AND Gender is NOT NULL
WITH CHECK OPTION;
```

The With Check Option alerts a database to the possibility that some insertions will be invalid!

```
INSERT INTO Fictional_Entities
VALUES ("Cthulhu", null,
"Hellboy");
```

# We Must Tell SQL to Watch for Infractions

# But Why Use "Views" At All?

Views are **virtual tables** that allow us to do the following:

- Structure data in a way that DB users or classes of users find more natural and intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need BUT no more.
- Synthesize (Join) and Summarize data from diverse tables into a single coherent “virtual” table.
- Generate reports based on these synthetic tables.

# When a Viewpoint is no longer useful for users ...

```
DROP VIEW Fictional_Entities;
```

```
DROP VIEW Real_Entities;
```

# We can simply DROP it like any other Table