

Database – Slide 8

Defining a Database Schema

CREATE TABLE name (list of elements).

- Principal elements are attributes and their types, but key declarations and constraints also appear.
- Similar CREATE *X* commands for other schema elements *X*: views, indexes, assertions, triggers.
- “DROP *X* name” deletes the created element of kind *X* with that name.

Example

```
CREATE TABLE Sells (  
    bar CHAR(20),  
    beer VARCHAR(20),  
    price REAL  
);
```

```
DROP TABLE Sells;
```

Types

- 1 . INT or INTEGER.
- 2 . REAL or FLOAT.
- 3 . CHAR (n) = fixed length character string, padded with “pad characters.”
- 4 . VARCHAR(n) = variable-length strings up to n characters.
 - ◆ Oracle uses VARCHAR2(n) as well. PostgreSQL uses VARCHAR and does not support VARCHAR2.

1. NUMERIC (*precision, decimal*) is a number with *precision* digits with the decimal point *decimal* digits from the right. NUMERIC (10 , 2) can store $\pm 99,999,999.99$
2. DATE. SQL form is DATE 'yyyy-mm-dd'
 - PostgreSQL follows the standard. Oracle uses a different format.
3. TIME. Form is TIME 'hh:mm:ss[.ss...]' in SQL.
4. DATETIME or TIMESTAMP. Form is TIMESTAMP 'yyyy-mm-dd hh:mm:ss[.ss...]' in SQL.
5. INTERVAL. Form is INTERVAL 'n *period*' in PostgreSQL. *Period* is month, days, year, etc.

PostgreSQL Dates

PostgreSQL supports extensive date calculations.

- Conversions `to_date(text)`, `to_char(date/time/etc.)`, `interval(text)`
- $\text{Date} \pm \text{Integer} = \text{Date}$;
 $\text{Date} - \text{Date} = \text{Integer}$ (always = number of days);
 $\text{Date} + \text{Date}$ is invalid!
- $\text{Timestamp} \pm \text{Interval} = \text{Timestamp}$;
 $\text{Timestamp} - \text{Timestamp} = \text{Interval}$;
 $\text{Interval} \pm \text{Interval} = \text{Interval}$;
 $\text{Date} + \text{Date}$ is invalid.
- Interval: '1 month' could be 28, 29, 30, or 31 days;
'31 days' is always just that.
- SQL uses `DATEADD` and `DATEDIFF`;
PostgreSQL uses the simpler `+` and `-`.
- Also `CURRENT_DATE`, `CURRENT_TIME`, `CURRENT_TIMESTAMP`.

Declaring Keys

Use `PRIMARY KEY` or `UNIQUE`.

- But only one primary key, many `UNIQUE`s allowed.
- SQL permits implementations to create an *index* (data structure to speed access given a key value) in response to `PRIMARY KEY` only.
 - ◆ But PostgreSQL and Oracle create indexes for both.
- SQL does not allow nulls in primary key, but allows them in “unique” columns (which may have two or more nulls, but not repeated non-null values).

Declaring Keys

Two places to declare:

2. After an attribute's type, if the attribute is a key by itself.
3. As a separate element.
 - ◆ Essential if key is >1 attribute.

Example

```
CREATE TABLE Sells (  
    bar CHAR(20),  
    beer VARCHAR(20),  
    price REAL,  
    PRIMARY KEY(bar,beer)  
);
```

Example

```
CREATE TABLE Sells (  
    bar CHAR(20),  
    beer VARCHAR(20),  
    price REAL,  
    UNIQUE(bar,beer)  
);
```

is different than:

```
CREATE TABLE Sells (  
    bar CHAR(20) UNIQUE,  
    beer VARCHAR(20) UNIQUE,  
    price REAL  
);
```

Other Properties You Can Give to Attributes

1. NOT NULL = every tuple must have a real value for this attribute.
2. DEFAULT value = a value to use whenever no other value of this attribute is known.

Example

```
CREATE TABLE Drinkers (  
    name CHAR(30) PRIMARY KEY,  
    addr CHAR(50)  
        DEFAULT '123 Sesame St',  
    phone CHAR(16)  
);
```

```
INSERT INTO Drinkers(name)
VALUES('Sally')
```

results in the following tuple:

name	addr	phone
Sally	123 Sesame St.	NULL

- Primary key is by default not NULL.
- This insert is legal.
 - ◆ OK to list a subset of the attributes and values for only this subset.
- But if we had declared
 phone CHAR(16) NOT NULL
then the insertion could not be made.

Interesting Defaults

- `DEFAULT CURRENT_TIMESTAMP`
- `SEQUENCE`

```
CREATE SEQUENCE customer_seq;  
CREATE TABLE Customer (  
    customerID INTEGER  
        DEFAULT nextval('customer_seq'),  
    name VARCHAR(30)  
);
```

Changing Columns

Add an attribute of relation R with

```
ALTER TABLE  $R$  ADD <column declaration>;
```

Example

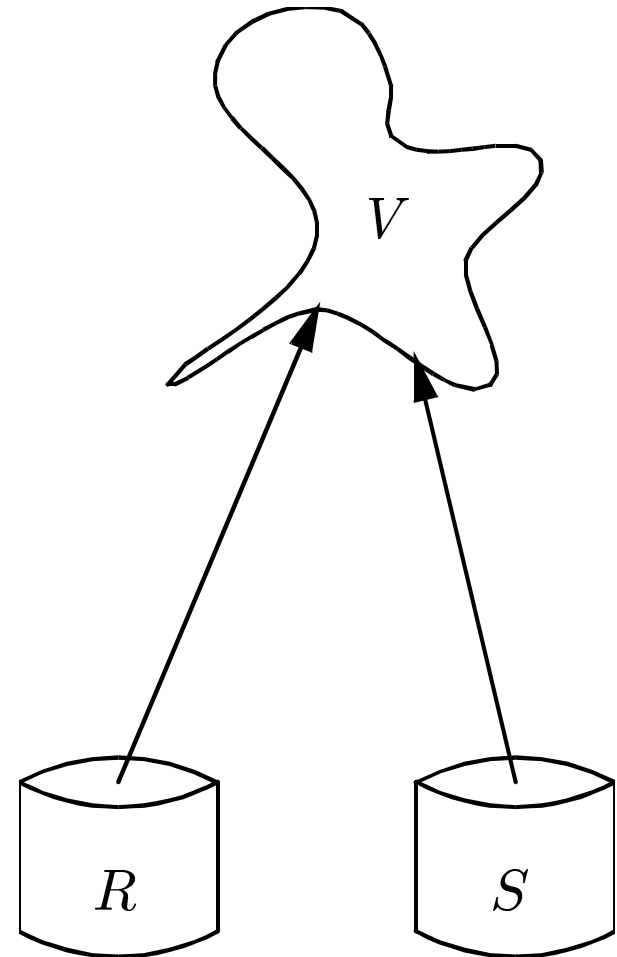
```
ALTER TABLE Bars ADD phone CHAR(16)  
    DEFAULT 'unlisted';
```

- Columns may also be dropped.

```
ALTER TABLE Bars DROP license;
```

Views

An expression that describes a table without creating it.



- View definition form is:

```
CREATE VIEW <name> AS <query>;
```

Example

The view `CanDrink` is the set of drinker-beer pairs such that the drinker frequents at least one bar that serves the beer.

```
CREATE VIEW CanDrink AS
  SELECT drinker, beer
  FROM Frequents, Sells
  WHERE Frequents.bar = Sells.bar;
```

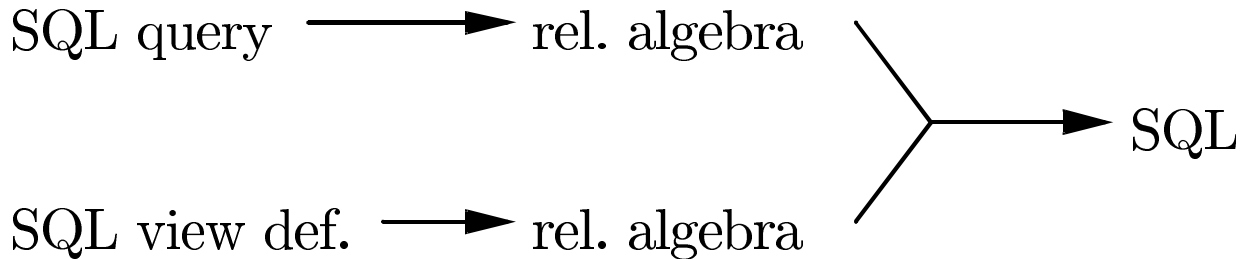
Querying Views

Treat the view as if it were a materialized relation.

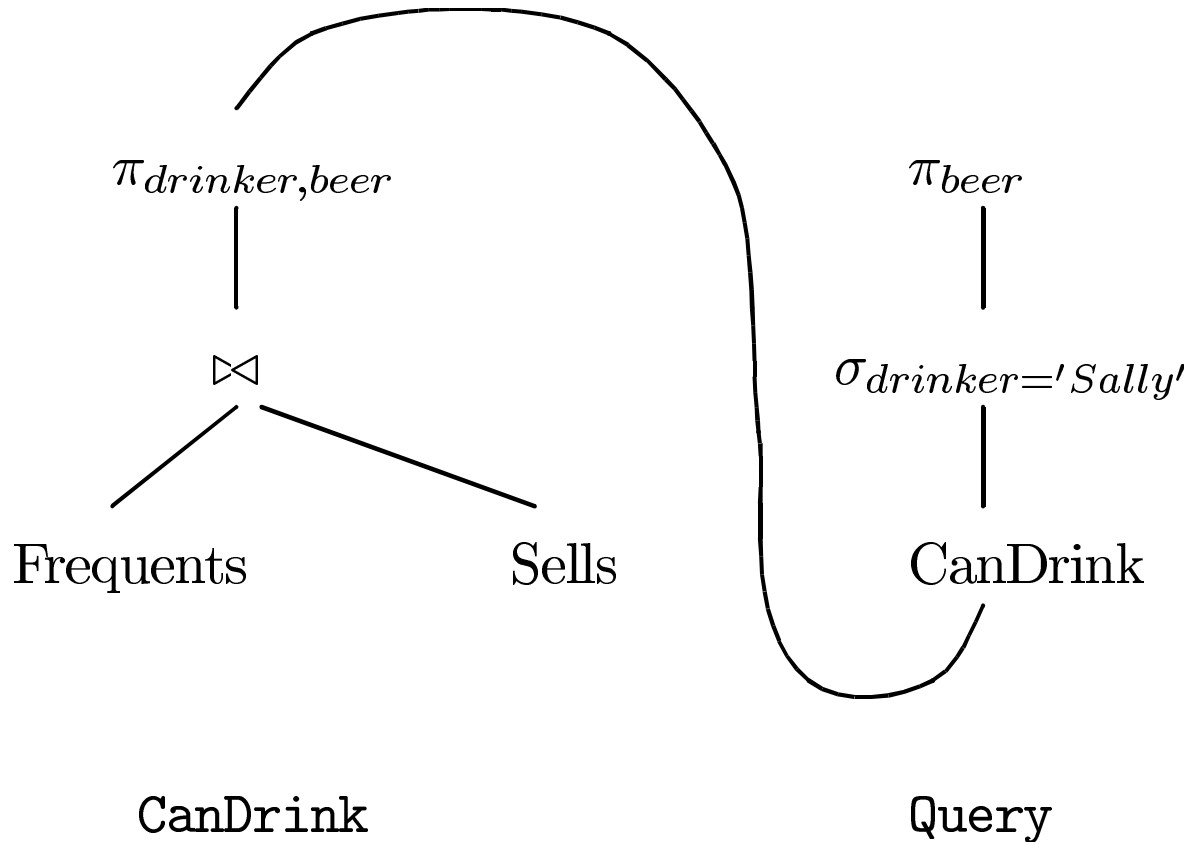
Example

```
SELECT beer
FROM CanDrink
WHERE drinker = 'Sally';
```

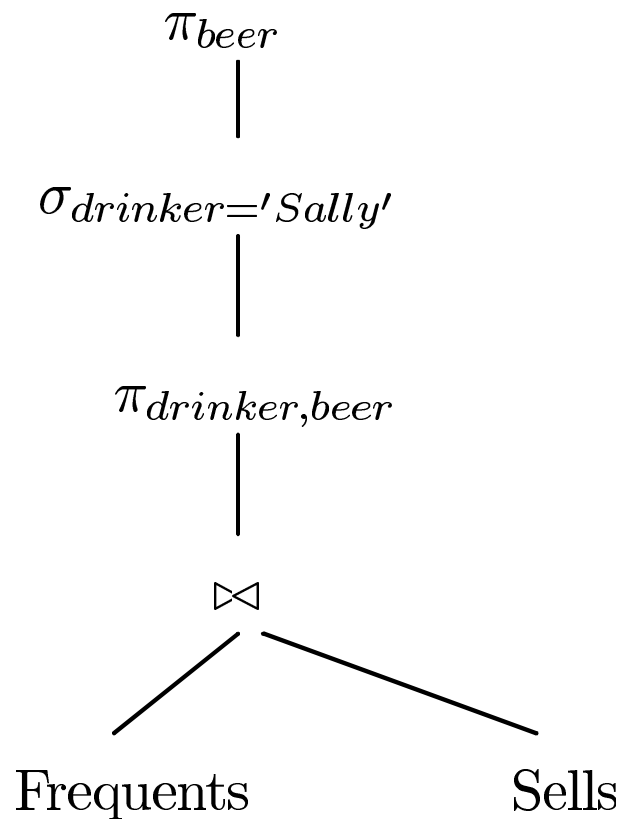
Semantics of View Use



Example



Compose



Optimize Query

1. Push selections down tree.
2. Eliminate unnecessary projections.

