Part II

Relational Databases – Data as Tables

Relational Databases – Data as Tables

Relational Databases – Data as Tables

1. Relations for Tabular Data

2. SQL Data Definition

Relational Databases – Data as Tables

1. Relations for Tabular Data
2. SQL Data Definition
3. Basic Operations: The Relational Algebra

Relational Databases – Data as Tables

1. Relations for Tabular Data
2. SQL Data Definition
3. Basic Operations: The Relational Algebra
4. SQL as a Query Language

Relational Databases – Data as Tables

1. Relations for Tabular Data
2. SQL Data Definition
3. Basic Operations: The Relational Algebra
4. SQL as a Query Language
5. Manipulation Operations in SQL

Educational Objective for Today . . .

- Basic understanding of the structure of relational databases

Educational Objective for Today . . .

- Basic understanding of the structure of relational databases
- Knowledge of base operations of relational query languages

Educational Objective for Today . . .

- Basic understanding of the structure of relational databases
- Knowledge of base operations of relational query languages
- Elementary ability to use SQL

Relations for Tabular Data

Relational Model

- Conceptually, a database is a set of tables

WINES

WineID	Name	Color	Vintage	Vineyard
1042	La Rose Grand Cru	Red	1998	Château La Rose
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
3478	Pinot Noir	Red	1999	Helena
4711	Riesling Reserve	White	1999	Müller
4961	Chardonnay	White	2002	Bighorn

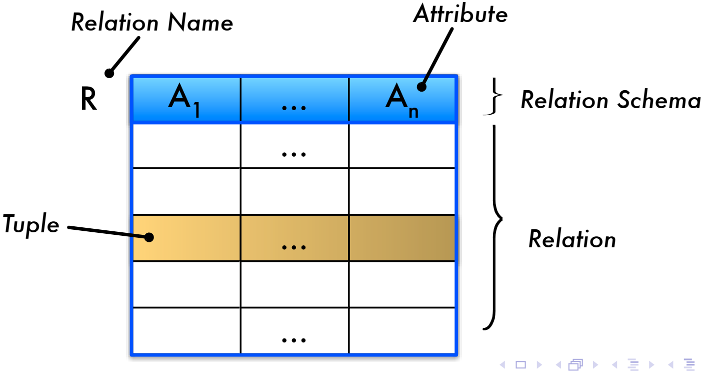
ORIGIN

Vineyard	District	Region
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	California

- Table = “Relation”

Presentation of Relations; Terminology

- **Bold** fields: relation schema
- Further entries in the table: relation
- A table row: tuple
- A column heading: attribute
- An entry: attribute value



Integrity Constraints: Keys

- Attributes of a column unambiguously identify stored tuples: key property
- E.g., **Vineyard** for table **ORIGIN**

ORIGIN

Vineyard	District	Region
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	California

- Combinations of attributes can also be keys!
- Keys can be marked by underlining them

Integrity Constraints: Foreign Keys

- Keys in one table can be used as unambiguous references in another table (or even in the same table!): Foreign key, referential integrity
- E.g., **Vineyard** as a reference to **ORIGIN**
- A foreign key is a key in a “foreign” table

Foreign Keys /2

WINES

WineID	Name	Color	Vintage	Vineyard → ORIGIN
1042	La Rose Grand Cru	Red	1998	Château La Rose
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
3478	Pinot Noir	Red	1999	Helena
4711	Riesling Reserve	White	1999	Müller
4961	Chardonnay	White	2002	Bighorn

ORIGIN

Vineyard	District	Region
Creek	Barossa Valley	South Australia
Helena	Napa Valley	California
Château La Rose	Saint-Emilion	Bordeaux
Château La Pointe	Pomerol	Bordeaux
Müller	Rheingau	Hessen
Bighorn	Napa Valley	California

SQL Data Definition

The **create table** Statement

```
create table base_relation_name (
             column_name$_1$ domain$_1$ [not null],
             ...
             column_name$_k$ domain$_k$ [not null])
```

- Effect of this command is both
  - to store the relation schema in the data dictionary, and
  - to prepare an “empty base relation” in the database

Possible Domains in SQL

- **`integer`** (also: **`integer4`**, **`int`**),
- **`smallint`** (also: **`integer2`**),
- **`float`**(*p*) (also, for short, **`float`**),
- **`decimal`**(*p*,*q*) and **`numeric`**(*p*,*q*) with *q* decimal places,
- **`character`**(*n*) (also, for short, **`char`**(*n*), with $n = 1$ just **`char`**) for character strings of fixed length *n*,
- **`character varying`**(*n*) (also, for short, **`varchar`**(*n*) for variable-length character strings up to the maximum length *n*,
- **`bit`**(*n*) or **`bit varying`**(*n*) like **`varchar`** but for bit strings, and
- **`date`**, **`time`**, **`timestamp`** for specifying dates, times and the combination of date and time

Example for **create table**

```
create table WINES (
   WineID int not null,
   Name varchar(20) not null,
   Color varchar(10),
   Vintage int,
   Vineyard varchar(20),
   primary key(WineID))
```

- **primary key** marks column as key attribute

create table with Foreign Key

```
create table WINES (
   WineID int,
   Name varchar(20) not null,
   Color varchar(10),
   Vintage int,
   Vineyard varchar(20),
   primary key(WineID),
   foreign key(Vineyard)
      references ORIGIN(Vineyard))
```

- **foreign key** marks column as a foreign key

Null Values

- **`not null`** precludes null values as attribute values for certain columns
- SQL uses **`null`** to refer to null values; we use $\perp$
- **`null`** has the semantics of "*unknown value*", "*value does not apply*" oder "*value does not exist*"; however, **`null`** itself does not belong to any domain
- **`null`** can occur in any column, except for key attributes or columns marked **`not null`**

Additional Notes on Data Definition in SQL

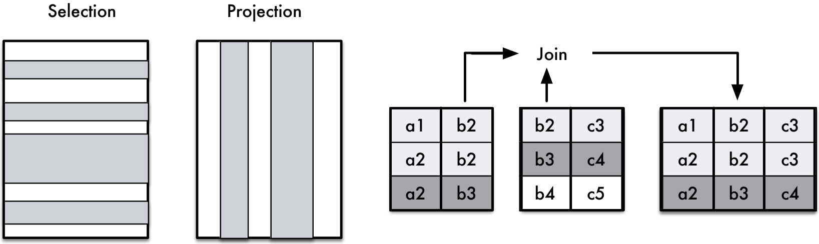
- Apart from primary and foreign keys, SQL allows specifying the following:
  - Default values for attributes using the **`default`** clause,
  - **`create domain`** statement to define custom domains (data types), and
  - **`check`** clause to specify further local integrity constraints within the domains, attributes and relation schemata being defined

Basic Operations: The Relational Algebra

Query Operations on Tables

- Basic operations on tables that allow computing new result tables from saved database tables
- Operations are combined to form the so-called relational algebra
- Mathematics: algebra is defined by a domain and operations defined on that domain
  → for database queries, the contents of the database are the values (of the domain), operations are functions to compute query results
- Query operations can be freely combined and form an algebra to perform “calculations on tables” – the so-called relational algebra

Relational Algebra: Overview



a1	b2
a2	b2
a2	b3

b2	c3
b3	c4
b4	c5

a1	b2	c3
a2	b2	c3
a2	b3	c4

Selection σ

- Selection: Choose rows in a table based on a selection predicate

$$\sigma_{\texttt{Vintage}>2000}(\texttt{WINES})$$

WineID	Name	Color	Vintage	Vineyard
2168	Creek Shiraz	Red	2003	Creek
3456	Zinfandel	Red	2004	Helena
2171	Pinot Noir	Red	2001	Creek
4961	Chardonnay	White	2002	Bighorn

Projection π

- Projection: Choose columns by specifying a list of attributes

$$\pi_{\texttt{Region}}(\texttt{ORIGIN})$$

Region
South Australia
California
Bordeaux
Hessen

Projection π

- Projection: Choose columns by specifying a list of attributes

$$\pi_{\texttt{Region}}(\texttt{ORIGIN})$$

Region
South Australia
California
Bordeaux
Hessen

- Projection removes duplicate tuples.

Natural Join ⋈

- Join: connects tables via same-named columns, combining two tuples if they have equal values in those columns

`WINES ⋈ ORIGIN`

WineID	Name	...	Vineyard	District	Region
1042	La Rose Grand Cru	...	Ch. La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	South Australia
3456	Zinfandel	...	Helena	Napa Valley	California
2171	Pinot Noir	...	Creek	Barossa Valley	South Australia
3478	Pinot Noir	...	Helena	Napa Valley	California
4711	Riesling Reserve	...	Müller	Rheingau	Hessen
4961	Chardonnay	...	Bighorn	Napa Valley	California

Natural Join $\bowtie$

- Join: connects tables via same-named columns, combining two tuples if they have equal values in those columns

$$\texttt{WINES} \bowtie \texttt{ORIGIN}$$

WineID	Name	...	Vineyard	District	Region
1042	La Rose Grand Cru	...	Ch. La Rose	Saint-Emilion	Bordeaux
2168	Creek Shiraz	...	Creek	Barossa Valley	South Australia
3456	Zinfandel	...	Helena	Napa Valley	California
2171	Pinot Noir	...	Creek	Barossa Valley	South Australia
3478	Pinot Noir	...	Helena	Napa Valley	California
4711	Riesling Reserve	...	Müller	Rheingau	Hessen
4961	Chardonnay	...	Bighorn	Napa Valley	California

- The vineyard “Château La Pointe” is missing from the result ⇝ tuples that do not find a partner (*dangling tuples*), are eliminated

Combining Operations

$$\pi_{\text{Name,Color,Vineyard}}\big(\sigma_{\text{Vintage}>2000}(\texttt{WINES}) \bowtie \sigma_{\text{Region='California'}}(\texttt{ORIGIN})\big)$$

yields

Name	**Color**	**Vineyard**
Zinfandel	Red	Helena
Chardonnay	White	Bighorn

Renaming β

- Renaming to adapt attribute names:

WINELIST

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve

RECOMMENDATION

Wine
La Rose Grand Cru
Riesling Reserve
Merlot Selection
Sauvignon Blanc

- Adapt with:

$$\beta_{\mathrm{Name} \leftarrow \mathrm{Wine}}\ (\mathtt{RECOMMENDATION})$$

Set Operations

- Union $r_1 \cup r_2$ of two relations r_1 and r_2: collects the tuple sets of two relations in a common schema
- Both relations must have the same set of attributes

$$\mathtt{WINELIST} \cup \beta_{\mathtt{Name} \leftarrow \mathtt{Wine}}(\mathtt{RECOMMENDATION})$$

Name
La Rose Grand Cru
Creek Shiraz
Zinfandel
Pinot Noir
Riesling Reserve
Merlot Selection
Sauvignon Blanc

Set Operations /2

- Difference $r_1 - r_2$ removes from the first relation all tuples that are present in the second relation

$$\texttt{WINELIST} - \beta_{\texttt{Name}\leftarrow\texttt{Wine}}(\texttt{RECOMMENDATION})$$

yields:

Name
Creek Shiraz
Zinfandel
Pinot Noir

Set Operations /3

- Intersection $r_1 \cap r_2$: yields all tuples that are present in both relations

$$\texttt{WINELIST} \cap \beta_{\texttt{Name} \leftarrow \texttt{Wine}}(\texttt{RECOMMENDATION})$$

yields:

Name
La Rose Grand Cru
Riesling Reserve

SQL as a Query Language

SQL Query as a Standard Language

- Query a single table

```sql
select Name, Color
from WINES
where Vintage = 2002
```

- SQL has multi-set semantics — SQL does not automatically suppress duplicate table entries!
- Set semantics by using **distinct**

```sql
select distinct Name
from WINES
```

Joining Tables

- Cross join as basic join

```
select *
from WINES, ORIGIN
```

- Join with operator **natural join**

```
select *
from WINES natural join ORIGIN
```

- Alternatively, join by specifying a join condition!

```
select *
from WINES, ORIGIN
where WINES.Vineyard = ORIGIN.Vineyard
```

Combining Conditions

- Expression in relational algebra

$$\pi_{\text{Name,Color,Vineyard}}\left(\sigma_{\text{Vintage}>2000}(\texttt{WINES}) \bowtie \sigma_{\text{Region='California'}}(\texttt{ORIGIN})\right)$$

- Query in SQL

```
select Name, Color, WINES.Vineyard
from WINES, ORIGIN
where Vintage > 2000 and
      Region = 'California' and
      WINES.Vineyard = ORIGIN.Vineyard
```

Set Operations in SQL

- In SQL, union is realized by an extra operator, **union**
- Differences by using nested queries

```
select *
from WINEMAKER
where Name not in (
      select Surname
      from CRITIC)
```

Manipulation Operations in SQL

Manipulation Operations in SQL

- **`insert`**: Insert one or more tuples into a base relation or view
- **`update`**: Change one or more tuples in a base relation or view
- **`delete`**: Delete one or more tuples from a base relation or view
- Local and global integrity constraints must be checked automatically by the system when executing manipulation operations.

The **update** Statement

- Syntax:

```
update base_relation
set    attribute$_1$ = expression$_1$
       ...
       attribute$_n$ = expression$_n$
       [ where condition ]
```

Example for **update**

WINES

WineID	**Name**	**Color**	**Vintage**	**Vineyard**	**Price**
2168	Creek Shiraz	Red	2003	Creek	7.99
3456	Zinfandel	Red	2004	Helena	5.99
2171	Pinot Noir	Red	2001	Creek	10.99
3478	Pinot Noir	Red	1999	Helena	19.99
4711	Riesling Reserve	White	1999	Müller	14.99
4961	Chardonnay	White	2002	Bighorn	9.90

```sql
update WINES
set Price = Price * 1.10
where Vintage < 2000
```

Example for `update`: New Values

WINES

WineID	Name	Color	Vintage	Vineyard	Price
2168	Creek Shiraz	Red	2003	Creek	7.99
3456	Zinfandel	Red	2004	Helena	5.99
2171	Pinot Noir	Red	2001	Creek	10.99
3478	Pinot Noir	Red	1999	Helena	$\boxed{21.99}$
4711	Riesling Reserve	White	1999	Müller	$\boxed{16.49}$
4961	Chardonnay	White	2002	Bighorn	9.90

Additional Notes on **update**

- Operations on single tuples can be achieved by using the primary key:

```
update WINES
set Price = 7.99
where WineID = 3456
```

- Update the whole relation:

```
update WINES
set Price = 11
```

The `delete` Statement

- Syntax:

```
delete
from base_relation
[ where condition ]
```

- Delete a tuple from the `WINES` relation:

```
delete from WINES
where WineID = 4711
```

Additional Notes on **delete**

- Deletion of multiple tuples is the common case:

```
delete from WINES
where Color = 'White'
```

- Delete the whole relation:

```
delete from WINES
```

Additional Notes on **delete** /2

- Deletions can lead to violation of integrity constraints!
- Example: Violation of the foreign key property if there are still wines from this origin:

```
delete from ORIGIN
where District = 'Hessen'
```

The insert Statement

- Syntax:

```
insert
into base_relation
   [ (attribute$_1$, ..., attribute$_n$) ]
values (constant$_1$, ..., constant$_n$)
```

- Optional list of attributes allows for insertion of incomplete tuples

insert Examples

```
insert into ORIGIN (Vineyard, Region)
values ('Wairau Hills', 'Marlborough')
```

- Not all attributes given ⇝ Value of missing attribute `District` will be **null**

```
insert into ORIGIN
values ('Château Lafite', 'Medoc', 'Bordeaux')
```

Inserting Computed Data

- Syntax:

```
insert
into base_relation
       [ (attribute$_1$, ..., attribute$_n$) ]
   SQL-query
```

- Example:

```
insert into WINES (
   select ProdID, ProdName, 'Red', ProdYear,
      'Château Lafite'
   from SUPPLIER
   where SName = 'Aspri Spirits' )
```

Summary

- Relational model: database as a set of tables
- Integrity constraints in the relational model
- Table definition in SQL
- Relational algebra: query operators
- Basic concepts of SQL queries and manipulations

Control Questions

- What is a relation?

Control Questions

- What is a relation?
- What are the defining properties of the relational algebra?

Control Questions

- What is a relation?
- What are the defining properties of the relational algebra?
- How are objects from the real world represented in a relational database?

Control Questions

- What is a relation?
- What are the defining properties of the relational algebra?
- How are objects from the real world represented in a relational database?
- How can tables in SQL be defined and manipulated?

Control Questions

- What is a relation?
- What are the defining properties of the relational algebra?
- How are objects from the real world represented in a relational database?
- How can tables in SQL be defined and manipulated?
- What are integrity constraints?