Databases II

more queries
updates & integrity constraints

Gilles Falquet
Centre universitaire d'informatique
UniGE

Content

- Extensions to the basic operations
- Update operations
- Keys and foreign keys
- Data integrity

Aggregation operations (sum, avg, count, min, max)

- Compute values that depend on the whole set of selected rows of table

```
select aggregation-operation (column), ...
from ...
where ...
```

- Apply the selection operations (from ... where ...)
- Apply the the aggregation operation on the rows that have a non NULL value for this column

Examples

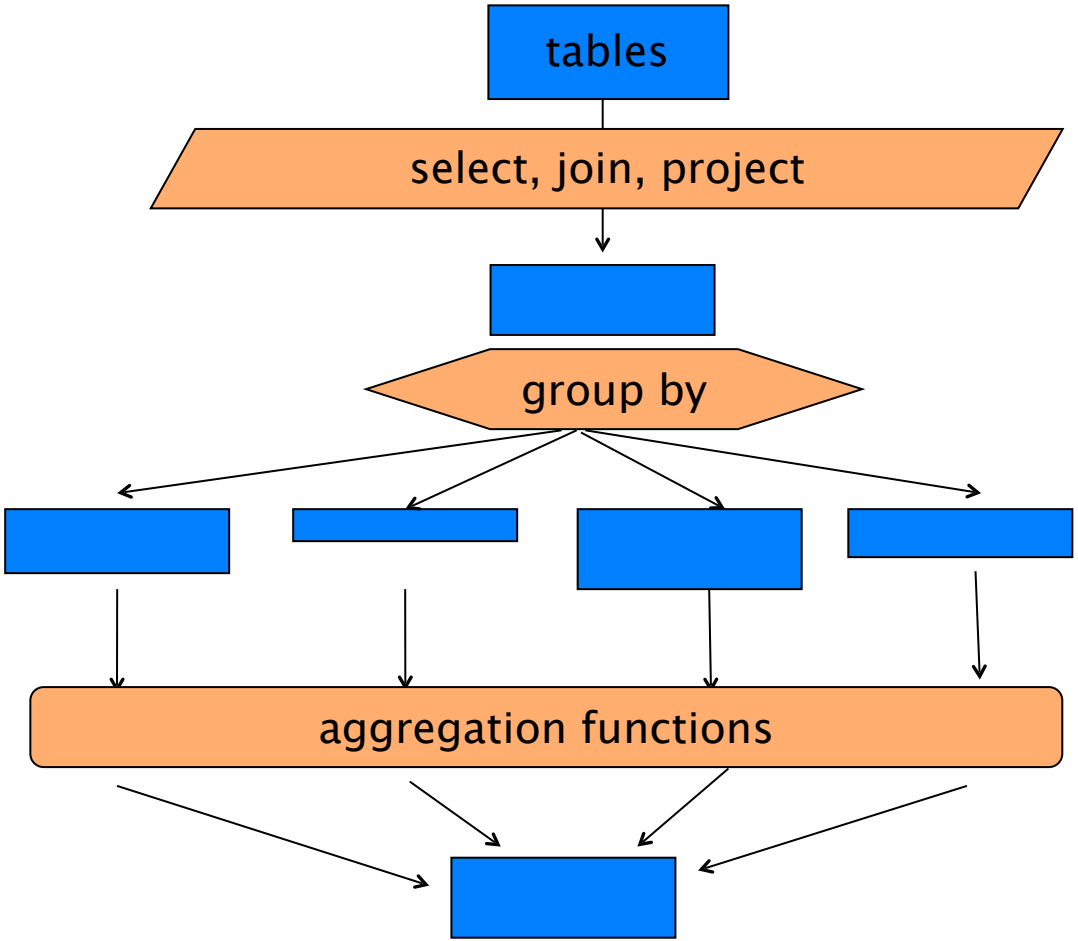
```
select avg(Price)
from Sales
where Date = 2033-11-05 and ItemType = 'bicycle’

select max(Price)
from Sales
where Date = 2033-11-05 and ItemType = 'bicycle'
```

```
SELECT count(city) FROM located
==>> 857

SELECT count(lake) FROM located
==>> 49

SELECT count(distinct lake) FROM located
==>> 27
```

City	Province	Country	River	Lake	Sea
Brazzaville	Congo	RCB	Zaire	NULL	NULL
Kinshasa	Kinshasa	ZRE	Zaire	NULL	NULL
Mbandaka	Equateur	ZRE	Zaire	NULL	NULL
Boma	Bas Zaire	ZRE	Zaire	NULL	NULL
Matadi	Bas Zaire	ZRE	Zaire	NULL	NULL
Whitehorse	Yukon Territory	CDN	Yukon River	NULL	NULL
Malakal	Aali an Nil	SUD	White Nile	NULL	NULL
Riga	Latvia	LV	Western Dwina	NULL	Baltic Sea
Bremen	Bremen	D	Weser	NULL	NULL
Bremerhaven	Bremen	D	Weser	NULL	North Sea

Grouping

- Make groups with the result of a select operation
- Criteria: groups tuples with the same value for an attribute (or expression)
- The attributes must be aggregated, except for the grouping attribute



```
select Region, sum(Quantity), min(Year)
from Sales
group by Region
```

Region	sum(Quantity)	min(Year)
CH	1181	2018
FR	132	2019

Sales

Quantity	Item	Year	Region
455	Car	2018	CH
112	Car	2019	FR
14	Bus	2019	CH
12	Bus	2020	FR
712	Car	2020	CH
9	Bus	2021	FR

Subqueries

- The selection part (where) of a query may contain a subquery
- The result of a subquery is either a single value or a multi-set of values

```
select Item
from Sales
where Quantity > (select avg(Quantity) from Sales)

select Item
from Sales
where Region in (select Region from World where Continent = 'Europe')
```

Subqueries and ANY

```
SELECT ... FROM T
WHERE A op ANY
    (SELECT B FROM T2 WHERE condition);
```

A tuple t is selected if there exist at least one value u in `(SELECT B FROM T2 WHERE condition)` such that

t.A op u is true.

```
SELECT name, population FROM city
WHERE population < any (SELECT population FROM city WHERE country = 'CH')
```

Subqueries and ALL

```
SELECT ... FROM T
WHERE A op ALL
    (SELECT B FROM T2 WHERE condition);
```

A tuple t is selected if for every value u in (SELECT B FROM T2 WHERE condition)

t.A op u is true

```
SELECT name, population FROM city
WHERE population > all (SELECT population FROM city WHERE country = 'CH')
```

NULL values and three-valued logic

- The domain of each attribute is extended with the special value NULL
- NULL may indicate that
  - the actual value is unknown
  - no value can exist (e.g. Weight for an eBook)
  - etc.

- Comparing NULL with another value always yields NULL (neither TRUE nor FALSE)

- The select operation selects the tuples for which the condition evaluates to TRUE

When $\sigma_C r \cup \sigma_{\text{not } C} r \neq r$

```
select * from Wine where Quality = Good
union
select * from Wine where Quality <> Good
```

Region	Year	Quality
Bordeaux	2010	Excellent
Valais	2014	Excellent
Bordeaux	2011	Good

Wine

Region	Year	Quality
Bordeaux	2010	Excellent
Bourgogne	2010	NULL
Valais	2014	Excellent
Valais	2013	NULL
Bordeaux	2011	Good

IS NULL and IS NOT NULL

- Test if a value is NULL (or not NULL)

`select * from Wine where Quality = NULL` ⟶ empty result

`select * from Wine where Quality IS NULL` ⟶

Region	Year	Quality
Bourgogne	2010	NULL
Valais	2013	NULL

3-valued logic

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

The relational algebra (and basic SQL) is not Turing complete

- Some functions cannot be computed in relational algebra
- The size of the result is always polynomial in the size of the relations
  - => impossible to compute exponential size results (powerset, permutations, etc.)
- Some relations cannot be expressed as a first-order logic formula on the database vocabulary
  - => they cannot be computed in SQL
  - e.g. the reachability in a graph

Example: reachability

The edges of a graph are represented in a relation

- Edge(source, destination, weight)

Find the nodes reachable from 'a' by following one edge

```
select e.destination from Edge e where e.source = 'a'
```

Find the nodes reachable from 'a' by following two edges

```
select e2.destination from Edge e1, Edge e2
where e1.source = 'a' and e1.destination = e2.source
```

Example: reachability (cont.)

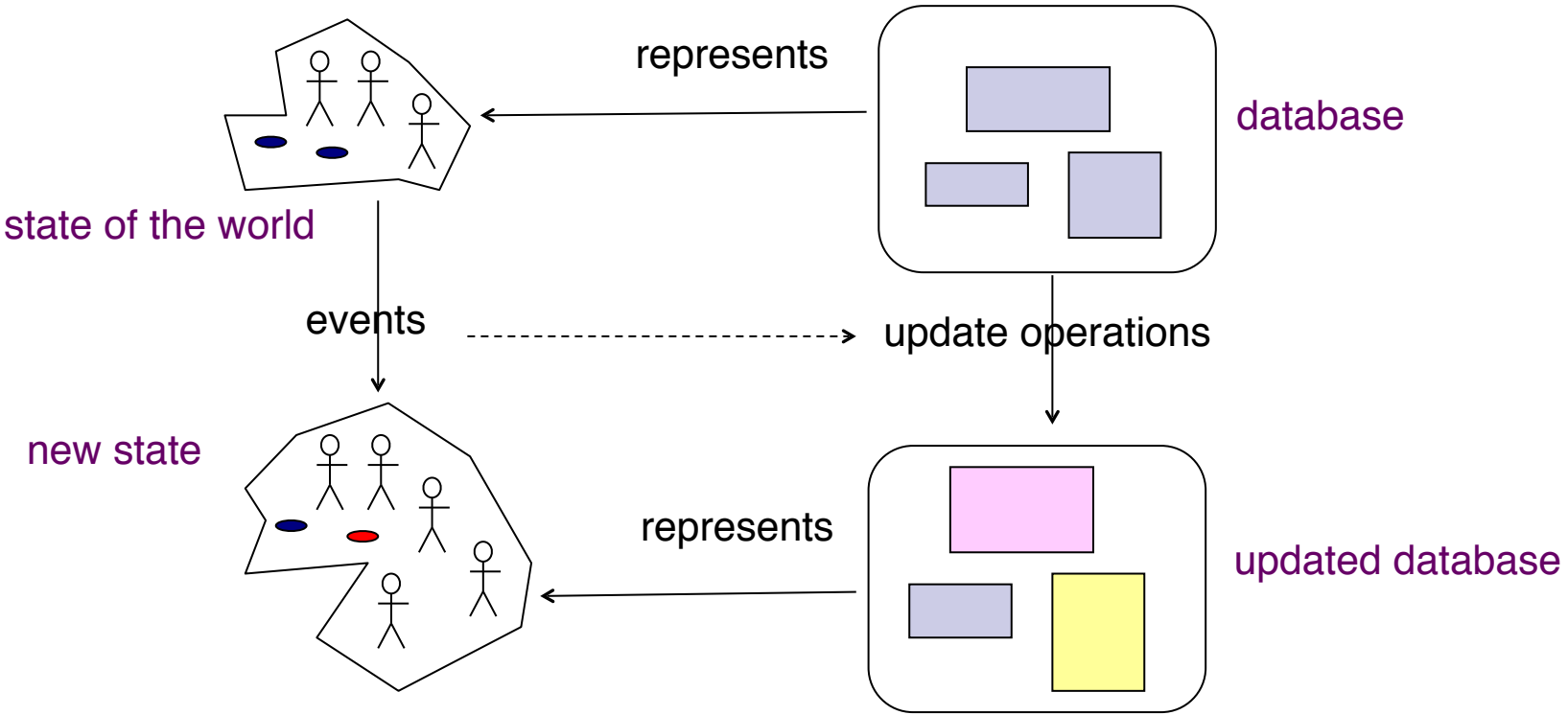
Find the nodes reachable from 'a' by following k edges

select e_k.destination from Edge e_1, Edge e_2, ..., Edge e_k
where e_1.source = 'a' and e_1.destination = e_2.source
and e_2.destination = e_3.source and ... ... and e_{k-1}. destination = e_k. source

Problem: we don't know in advance the number of edges to follow to reach every reachable node. It depends on the relation (data)

Rem. This is partially solved by additional “transitive closure” or recursive query operations provided by some DBMSs

Database updates



Insertion

insert into table(col_1, col_2, …, col_k)
values (val_1, val_2, …, val_k)

- Add a new row r with r.col_1 = val_1, ..., r.col_k = val_k, and r.c = NULL for the other columns

Example

```
insert into Wine(Region, Quality, Year)
       values ('Bourgogne", 'Good', 2022)
```

delete from table
    where condition

Remove all the rows that satisfy the condition

Exemple

```
delete from Wine where Region = 'Valais'
delete from Wine where Year < 2011
```

Modify one or more rows

update table
set $col_1 = val_1$, ..., $col_k = val_k$
where condition

For each row that satisfies the condition
    Modify the values for the columns col_k, ..., col_k

- Set the quality of the wines from Valais to Excellent for every year

```
update Wine set quality = 'Excellent'
where Region = 'Valais'
```

- The current values can be used to compute the new one

```
update Catalogue set Price= Price * 1.05
where Category = 'Toy'
```

Integrity Constraints

- Conditions that must remain true throughout the life of the database (invariants)
- Generally reflect general rules of the world, or specific rules of a domain in the data
  - « A latitude value must lie between -90 et +90 »
  - « At most one course may take place in a given room and time slice »
  - « Every car with a licence plate must have exactly one owner »
  - « A course may take place only if at least 4 students have enrolled »
- DBMSs can automatically check some categories of constraints

Constraint on a column value

- Restrict the set of accepted values in a column
  - forbid null values (NOT NULL constraint)
  - specify a subtype of the domain (e.g Strings with at most 8 characters)
  - specify a closed list of accepted values (ENUM type)

Key constraint

a set of attributes $A = \{A_1, ..., A_k\}$ is a key of relation schema R iff

for any relation instance r(R)

for any set of values $a_1, ..., a_k$

there is a most one tuple in r with $t.A_1 = a_1, ..., t.A_k = a_k$

- for a relation r the values of the key attributes determine the values of the other attributes

Examples

- Student(StudentNo, FirstName, LastName, EnrollData, Faculty)
  - there is only one student is a given number
- Wine(Region, Year, Quality, AvgPrice)
  - the quality and average price are determined by the region and year

```
CREATE TABLE Wine(Region VARCHAR, Year INT,
                  Quality VARCHAR, AvgPrice INT)
       PRIMARY KEY (Region, Year)
```

Foreign key constraint

- The value of (set of) column of a table T must be a key value of a (reference) table
- If K is a key of S and F is a foreign key of R with reference to S then for instance r(R) of R and s(S) of S
  - if $t \in r(R)$ and t.F is not NULL then there exists $u \in s(S)$ such that $t.F = u.K$

Example

Real-world constraint

- « A French university is located in a French departement. »

Database constraint

- The inDept column of the University table is a foreign key of the Departement

```
CREATE TABLE Departement(no INTEGER, population INTEGER, name VARCHAR
PRIMARY KEY ( no ))

CREATE TABLE Uni(nameVARCHAR, inDept INTEGER, …
FOREIGN KEY ( inDept) REFERENCES Departement ( no ))
```

University

name	inDept	….
PARIS VI	75	…
LYON III	204	…
PARIS X	75	…
U. Corse	20	…

Departement

no	population	name
1	634554	Ain
…		
20	323003	Corse
…		
75	5324443	Seine
…		
95		

Foreign key constraint violation

Foreign Keys and Relationships

- Foreign Keys express relationships between relations of a database
- These are n to 1 relationships
- Foreign keys ARE NOT relations
- Foreign keys and relations form a graph
- To represent an n to m relationship between R and S: add a relation

  R(KR, ...) ← T(FR, FS, ...) → S(KS, ...)

Team Scrabble Database Schema