Bancos de Dados Geográficos

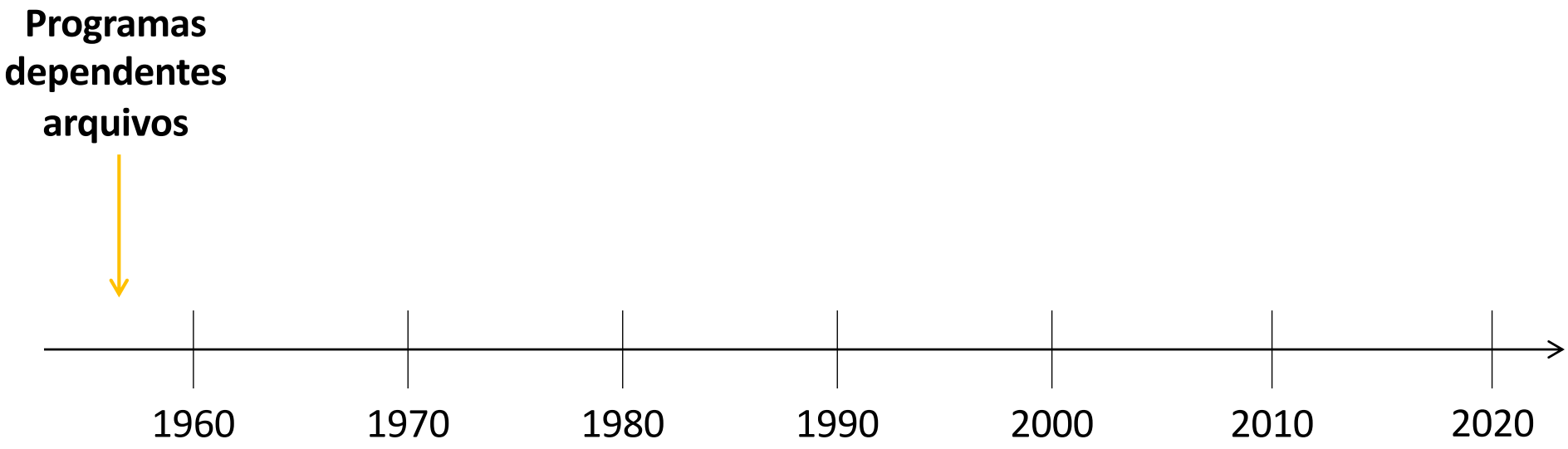
Curso de Verão - Geoinformática

27 de Janeiro de 2020

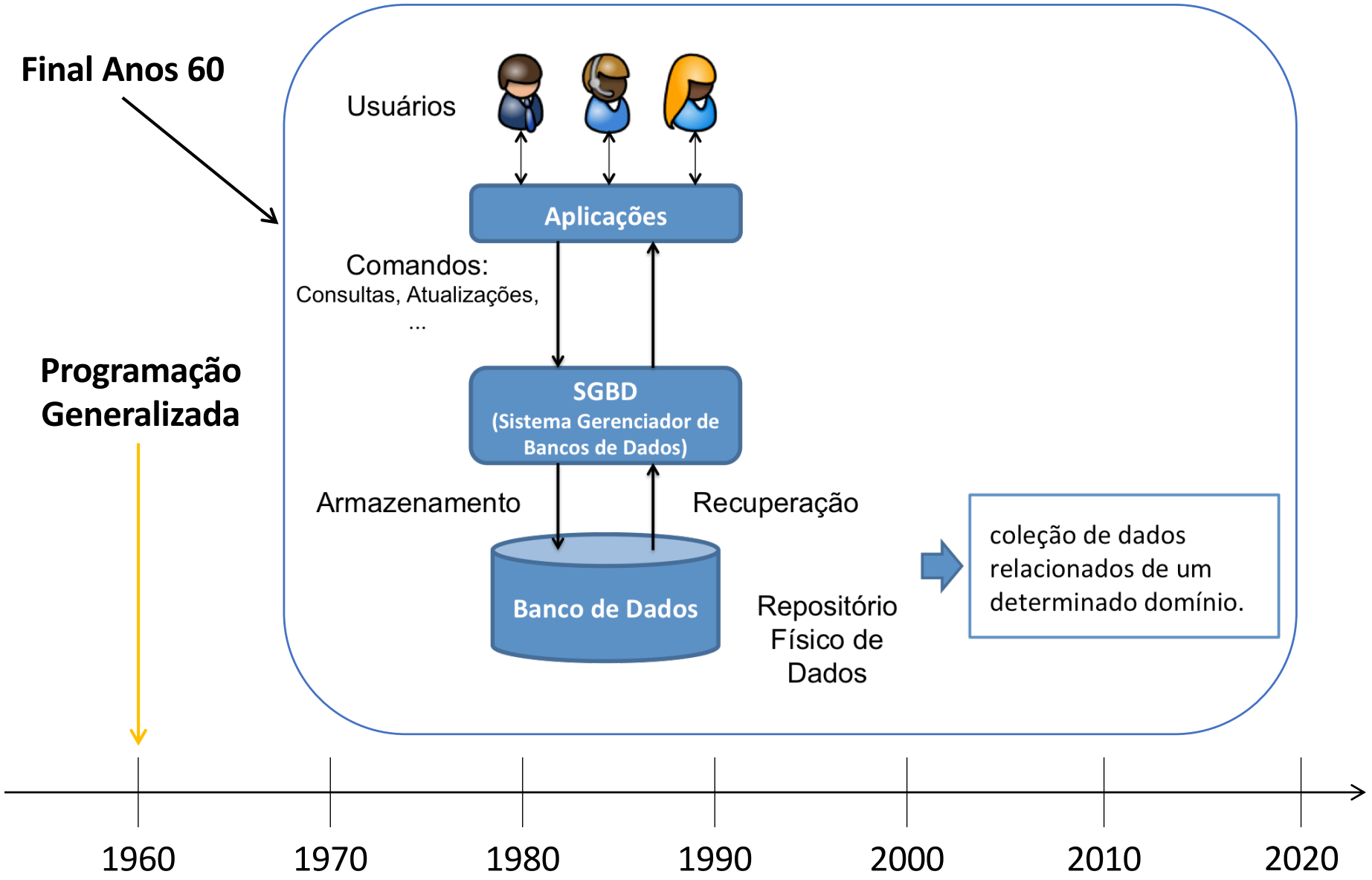
Evolução das Tecnologias de Bancos de Dados

Gilberto Ribeiro de Queiroz

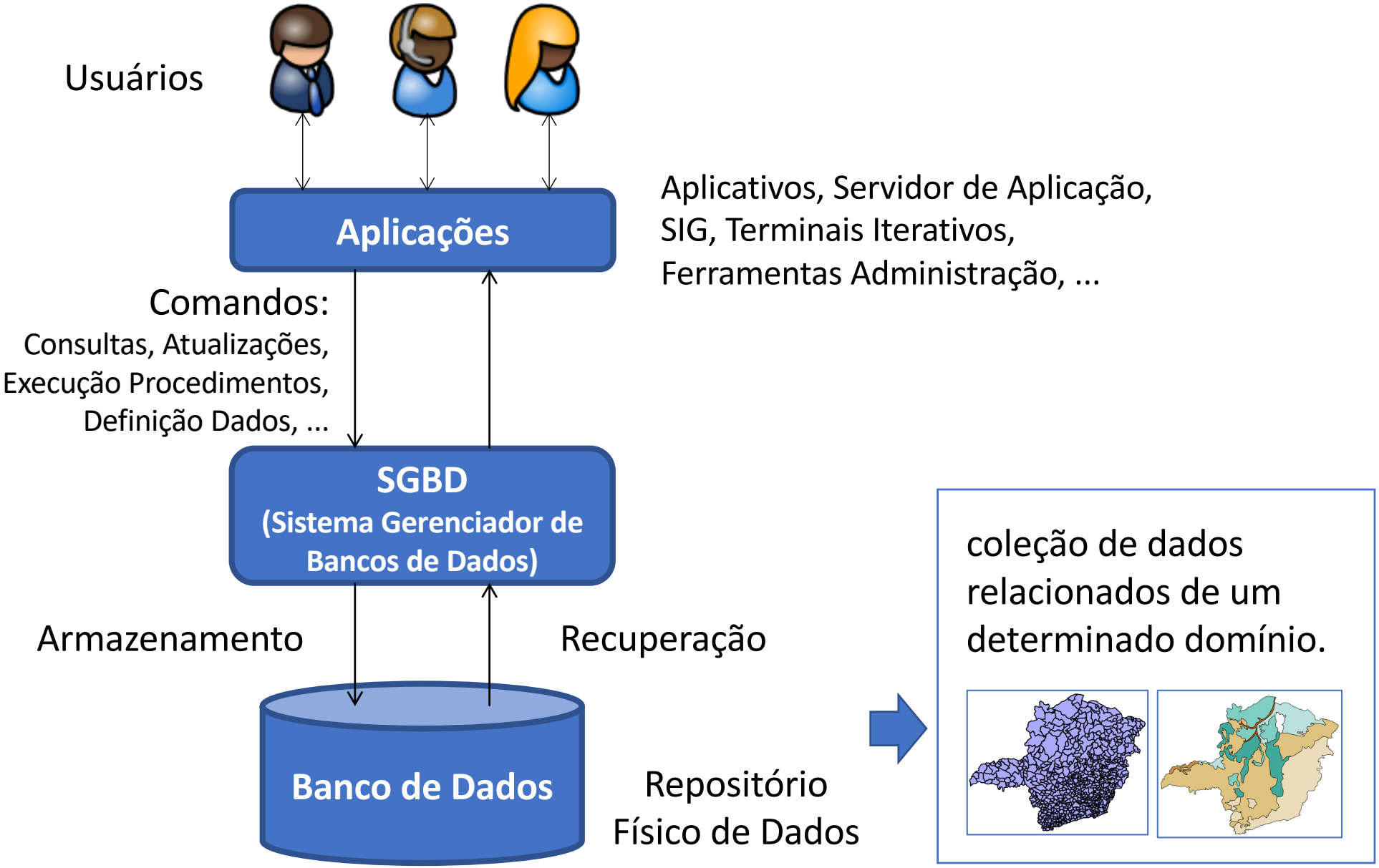
Evolução das Tecnologias de Bancos Dados



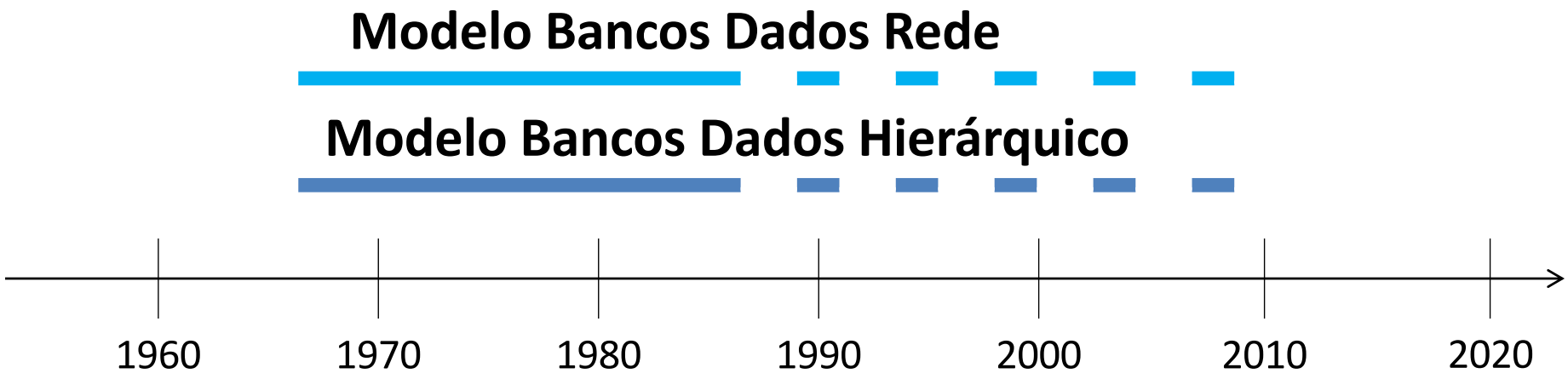
Evolução das Tecnologias de Bancos Dados



Sistemas de Bancos de Dados

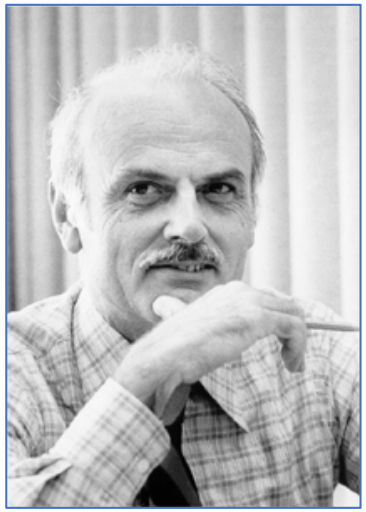


Evolução das Tecnologias de Bancos Dados



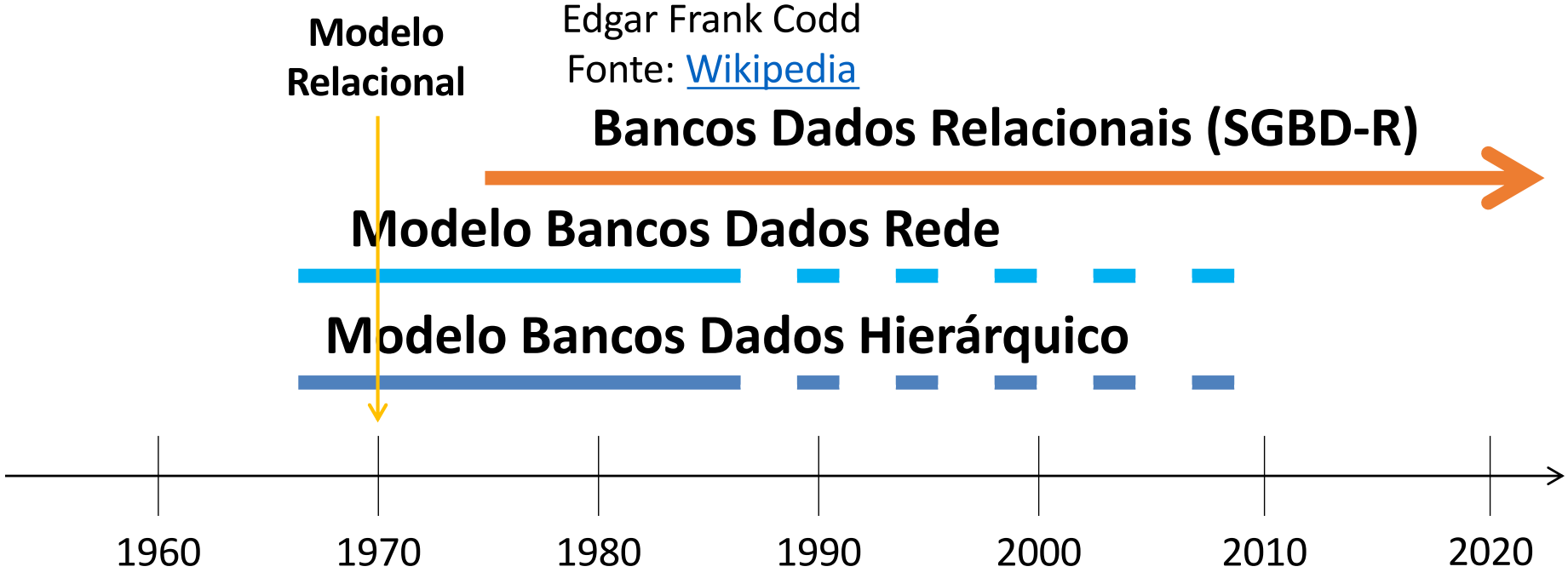
Evolução das Tecnologias de Bancos Dados

ACM Turing Award (1981)



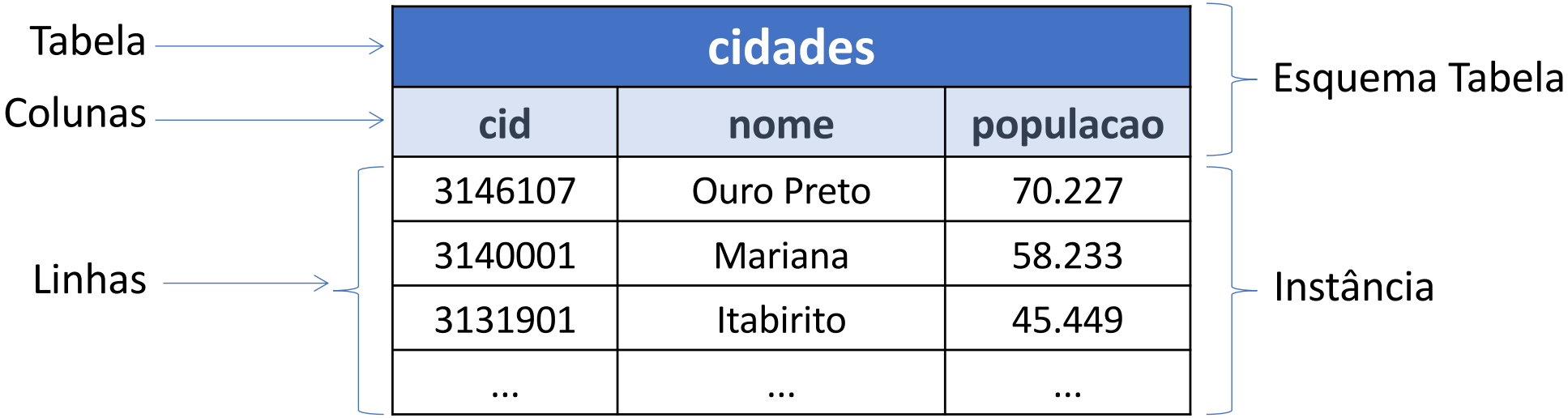
Edgar Frank Codd
Fonte: Wikipedia

E. F. Codd. 1970. ***A relational model of data for large shared data banks***. Communications of the ACM, v. 13, n. 6, June 1970, pp. 377-387.

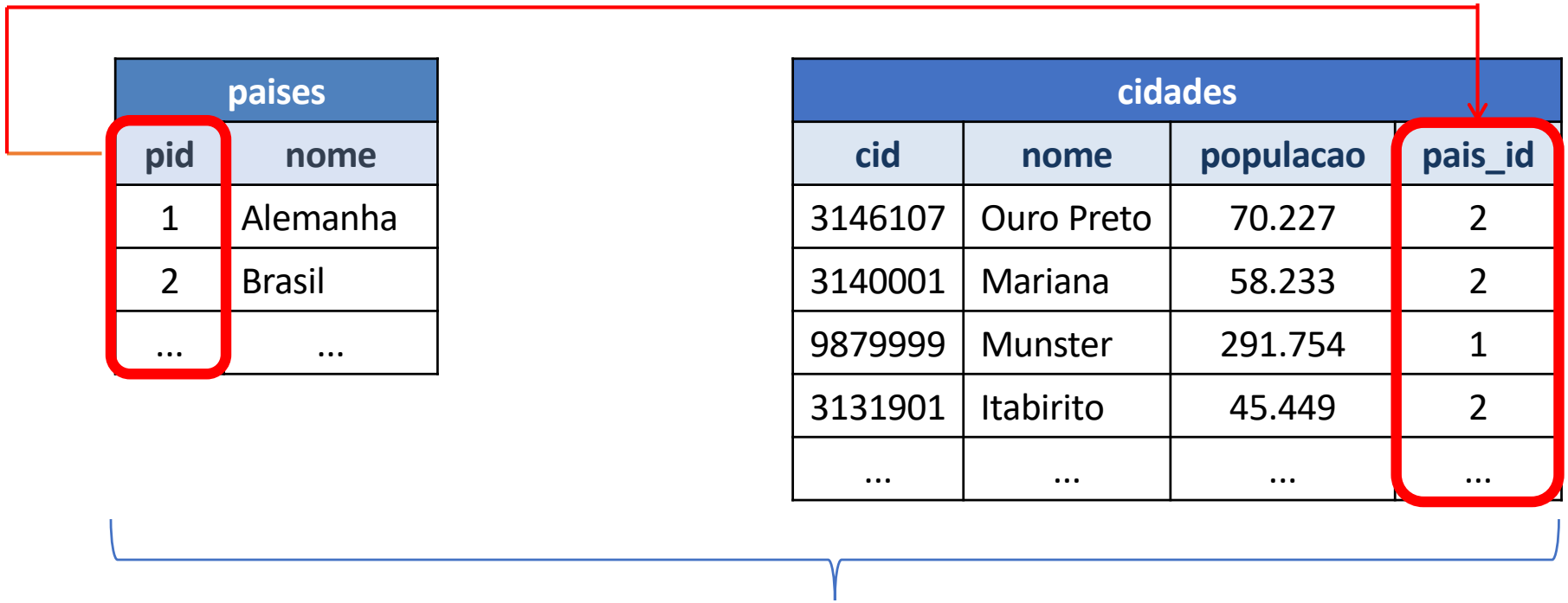


Relação (ou Tabela)

- Um banco de dados relacional é organizado em uma coleção de relações (ou tabelas) possivelmente relacionadas entre si.



Relacionamentos entre tabelas



paises	
pid	nome
1	Alemanha
2	Brasil
...	...

cidades			
cid	nome	populacao	pais_id
3146107	Ouro Preto	70.227	2
3140001	Mariana	58.233	2
9879999	Munster	291.754	1
3131901	Itabirito	45.449	2
...	...	...	...

paises_ x_cidades				
pid	p_nome	cid	c_nome	c_populacao
2	Brasil	3146107	Ouro Preto	70.227
2	Brasil	3140001	Mariana	58.233
1	Alemanha	9879999	Munster	291.754
2	Brasil	3131901	Itabirito	45.449
...		...	...	...

Álgebra Relacional

- **Exemplo:**
  - **Operador de Seleção:** seleciona tuplas de uma relação que satisfazem um certo predicado (ou condição).
  - **Consulta:** selecionar as cidades com população acima de 60.000 habitantes.

cidades			
cid	**nome**	**populacao**	**pais_id**
...	Ouro Preto	70.227	2
...	Mariana	58.233	2
...	Munster	291.754	1
...	Itabirito	45.449	2

Tabela/Relação de Entrada

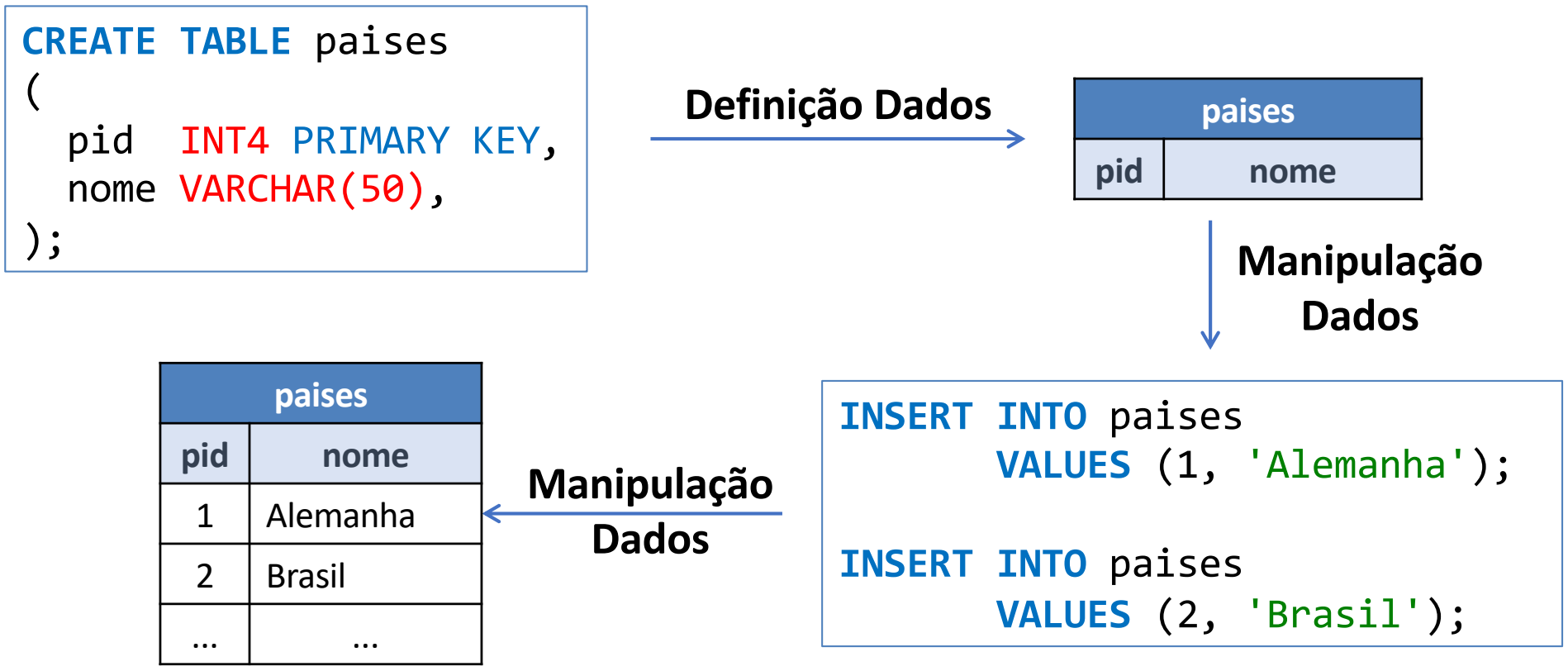
$$\sigma_{populacao>60000}(cidades)$$

nova_relação			
cid	**nome**	**populacao**	**pais_id**
...	Ouro Preto	70.227	2
...	Munster	291.754	1

Tabela/Relação de Saída

Linguagem de Consulta: SQL

- O modelo relacional é a base para linguagens de alto nível:
  - Álgebra/Cálculo Relacional → Linguagem Declarativa → ISO/SQL (Structured Query Language)



Linguagem de Consulta: SQL

- O modelo relacional é a base para linguagens de alto nível:
  - Álgebra/Cálculo Relacional → Linguagem Declarativa → ISO/SQL (Structured Query Language)

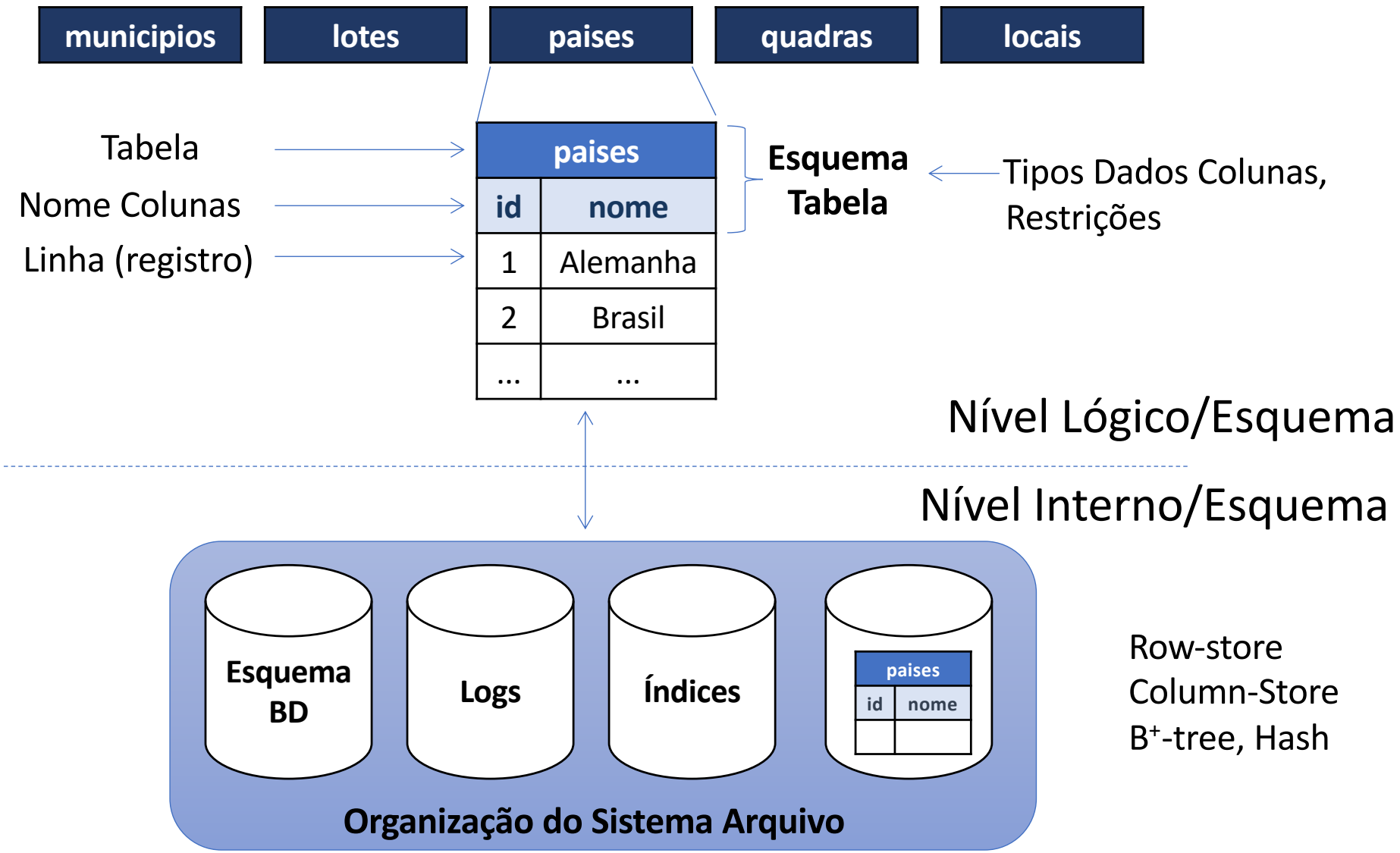
cidades			
cid	**nome**	**populacao**	**pais_id**
...	Ouro Preto	70.227	2
...	Mariana	58.233	2
...	Munster	291.754	1
...	Itabirito	45.449	2

```
SELECT nome
  FROM cidades
 WHERE populacao > 60000
```

Consulta (Não-Procedural)

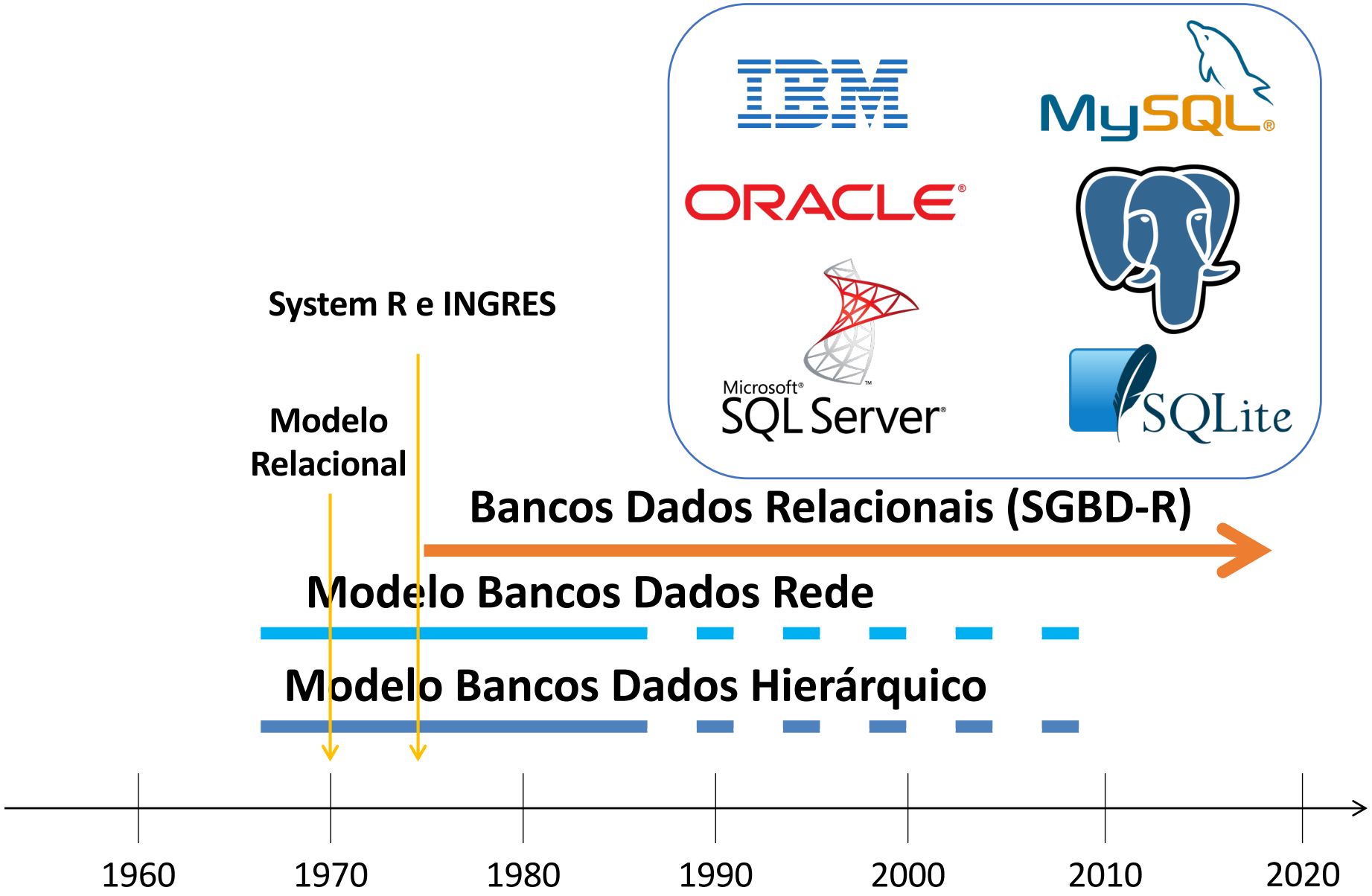
resultado
nome
Ouro Preto
Munster

Independência Física dos Dados



Fonte: Adaptado de Gray (1996)

Evolução das Tecnologias de Bancos Dados

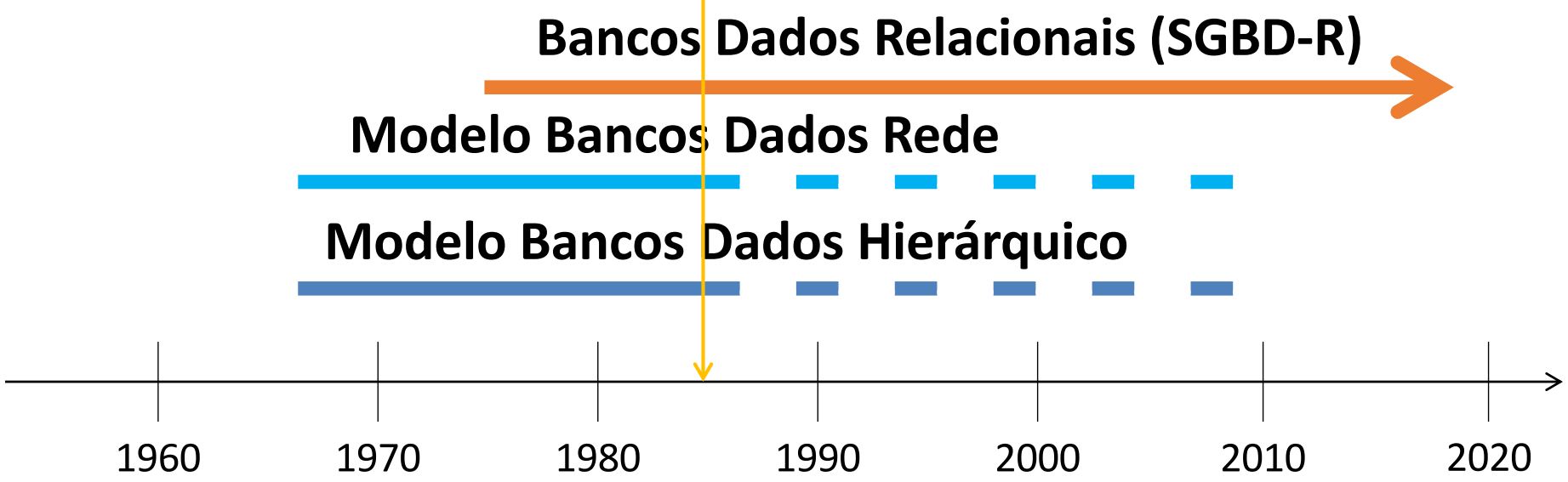


Evolução das Tecnologias de Bancos Dados

Aplicações emergentes e novas demandas:
CAD, SIG, Multimedia, OLAP, Real-time, Científicas

Período de muita pesquisa sobre extensibilidade dos SGBDs

Protótipos de pesquisa no final dos anos 80 voltados a SIG:
Probe, DASDBS GEO-Kernel, Gral, SIRO-DBMS, Starburst, Geo++, GéoSabrina, GODOT, $GeoO_2$, Paradise



Gral: An Extensible Relational Database System for Geometric Applications

Ralf Hartmut Güting

Fachbereich Informatik, Universität Dortmund
D-4600 Dortmund 50, West Germany

Abstract: We describe the architecture of a relational database system that is extensible by user-defined data types and operations, including relation operations. The central concept is to use languages based on many-sorted algebra to represent queries as well as query execution plans. This leads to a simple and clean extensible system architecture, eases the task of an application developer by providing a uniform framework, and also simplifies rule-based optimization. As a case study the extensions needed for a geometric database system are considered.

1. Introduction

Much of the database research of recent years was aimed at providing a better support for non-standard applications such as office information systems, geographic information systems, CAD databases, etc. A common need of these applications is the representation and manipulation of more complex objects than those representable by a tuple of a relation in the traditional relational model, for example, an office form, a complete map or a river, say, in a geographic information system, or the design of a VLSI circuit.

A fundamental choice for the representation of a complex

A lot of work has been done to support the modeling of visible object structures. Enhancements to the relational model have been proposed by linking together tuples to represent an object either explicitly [Co79, HaL82] or implicitly, through the use of nested relations [ScS86]. Most of the more recent data models and system proposals do also support structural object orientation, for example [MaD86, CaDV88, PiT86].

The idea of allowing application-specific abstract data types as base types, or attribute domains, of a database system was perhaps first put forward in [StRG83]. Since base types need to be implemented in a programming language and because they are application-specific, a user must be able to implement such a type and to add it to a database system. This observation has led to efforts by several groups to construct *extensible* database systems. Two directions can be distinguished. One is to select a data model and to implement for this data model a system with well-defined interfaces for user extensions. This is the approach chosen by the POSTGRES [StR86] and Starburst [Schw86] projects, based on the relational model, and within the PROBE project [Daya87] for an extended functional data model. A different view is taken in the EXODUS [Care86] and GENESIS [Bato86] projects where a collection of powerful tools for

R. H. Güting. **1989**. Gral: an extensible relational database system for geometric applications. In Proceedings of the 15th **international conference on Very large data bases (VLDB '89)**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 33-44.

The GEO++ System: an Extensible GIS

Tom Vijlbrief
TNO Institute for Perception,
P.O. Box 23, 3769 ZG Soesterberg, The Netherlands.
Email: tom@izf.tno.nl
and
Peter van Oosterom
TNO Physics and Electronics Laboratory,
P.O. Box 96864, 2509 JG The Hague, The Netherlands.
Email: oosterom@fel.tno.nl

In this paper we present a classification of the architectures of Geographic Information Systems (GIS). Most commercial GISs are closed. This means that if certain functionality is not available, it is impossible for the users to extend or modify the system for their own purpose. We then present a solution based on the extensible database management system "Postgres", in which new data types and operators may be defined. The resulting extensible GIS is called *GEO++*. We illustrate this powerful capability with two examples.

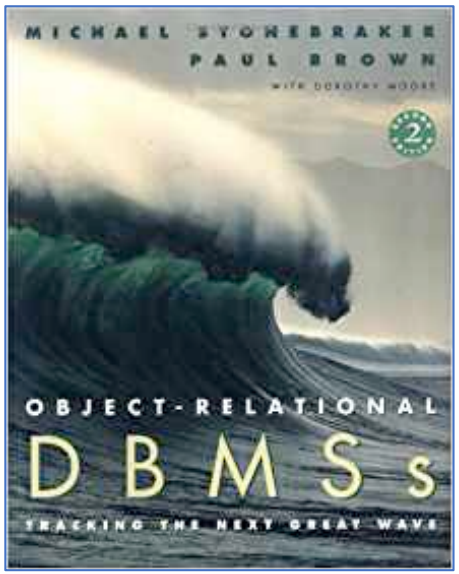
1 Introduction

Most commercial Geographic Information Systems (GISs) are based on a relational DataBase Management System (DBMS), such as Oracle or Ingres. One obvious drawback of the standard DBMSs is that they cannot manipulate geographic data. That is, there are no ge-

Rowe, & Hirohama, 1990) that we implemented. Section 4 enumerates the basic capabilities of our Postgres GIS front-end *GEO++*. The implementation of the system has been described in an earlier paper (van Oosterom & Vijlbrief, 1991). The real power of GEO++ is demonstrated in Section 5, in which the system is extended with user-defined types.

Vijlbrief, T. and van Oosterom, P. The GEO++ system: An extensible GIS. Proceedings of the Fifth **International Symposium on Spatial Data Handling**, Charleston, SC, **1992**.

Evolução das Tecnologias de Bancos Dados



Stonebraker et al. (1996)

PostgreSQL

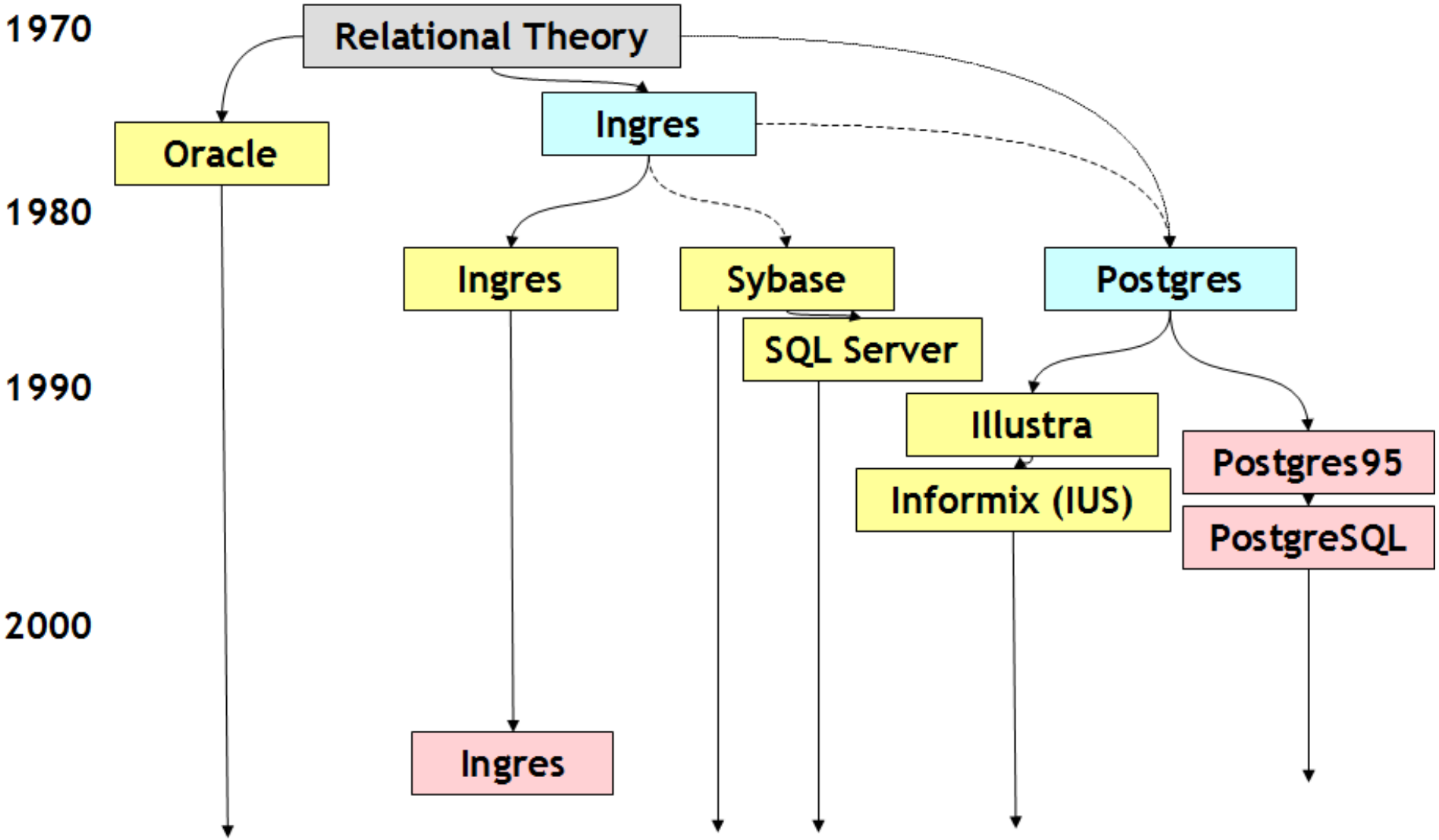
- É um sistema gerenciador de bancos de dados objeto-relacional open-source (licença ao estilo BSD):
  - http://www.postgresql.org

- O código fonte do seu núcleo encontra-s escrito na Linguagem de Programação C.

- Disponível para diversas plataformas: Linux, Mac OS X e Microsoft Windows.

- Versão que usaremos durante o curso: 11.x

História do PostgreSQL



Fonte: Ramsey (2007)

Prática

Criando e Consultando Tabelas no PostgreSQL

Sistemas Gerenciadores de Bancos de Dados Objeto-Relacional (SGBD-OR)

- Em um SGBD-OR, um **tipo de dado** (*data type*) é definido por uma **representação de armazenamento** juntamente com **operadores e funções** sobre este tipo.

- A flexibilidade do sistema de tipos dos SGBD-OR os tornam uma ferramenta poderosa para modelar aplicações mais complexas, tais como GIS.

SGBD-OR: User Defined Types

```
CREATE TYPE geo_point AS
(
   x     REAL,
   y     REAL,
   srid  INTEGER
);
```

O comando **CREATE TYPE** permite definir a **representação de armazenamento** para o **tipo geo_point**.

SGBD-OR: User Defined Types

```
CREATE TABLE escolas
(
  gid           INTEGER PRIMARY KEY,
  nome          VARCHAR(100)
  localizacao GEO_POINT
);
```

```
INSERT INTO escolas
     VALUES (3, 'Instituto São José',
                '(1, 2, 4326)'::GEO_POINT);
```

SGBD-OR: User Defined Functions

```
CREATE OR REPLACE FUNCTION distance(first GEO_POINT, second GEO_POINT)
RETURNS REAL
AS $$
DECLARE
  dx REAL;
  dy REAL;
BEGIN
  dx = (first.x - second.x);
  dy = (first.y - second.y);
  RETURN sqrt(dx * dx + dy * dy);
END;
$$
LANGUAGE plpgsql;
```

O comando **CREATE FUNCTION** permite **criar** ou **estender** a **álgebra de um determinado tipo de dado**, neste caso o tipo **geo_point**.

SGBD-OR: **U**ser **D**efined **F**unctions

```
SELECT distance('(1, 2, 4326)'::GEO_POINT, '(10, 20, 4326)'::GEO_POINT);
```

As funções definidas pelo usuário passam a fazer, automaticamente, parte da **linguagem de consulta do SGBD**.

SGBD-OR: Sobrecarga de Operadores

```
CREATE OR REPLACE FUNCTION less_than(first GEO_POINT, second GEO_POINT)
RETURNS REAL
AS $$
BEGIN
  IF(first.x < second.x)
  THEN RETURN TRUE;
  END IF;

  IF(first.x > second.x)
  THEN RETURN FALSE;
  END IF;
  ...
  RETURN FALSE;
END;
$$
LANGUAGE plpgsql;
```

SGBD-OR: Sobrecarga de Operadores

```
SELECT less_than('(1, 2, 4326)'::GEO_POINT, '(10, 20, 4326)'::GEO_POINT);

SELECT less_than('(1, 2, 4326)'::GEO_POINT, '(-1, 2, 4326)'::GEO_POINT);
```

SGBD-OR: Sobrecarga de Operadores

```
CREATE OPERATOR <
(
  leftarg = GEO_POINT,
  rightarg = GEO_POINT,
  procedure = less_than,
  commutator = >,
  negator = >=
);
```

```
SELECT '(1, 2, 4326)'::GEO_POINT < '(10, 2, 4326)'::GEO_POINT;
```

SGBD-OR: User Defined Access Methods

B-tree

Operation	Strategy Number
less than	1
less than or equal	2
equal	3
greater than or equal	4
greater than	5

Hash

Operation	Strategy Number
equal	1

GiST – Rtree 2D

Operation	Strategy Number
strictly left of	1
does not extend to right of	2
overlaps	3
does not extend to left of	4
strictly right of	5
same	6
contains	7
contained by	8
does not extend above	9
strictly below	10
strictly above	11
does not extend below	12

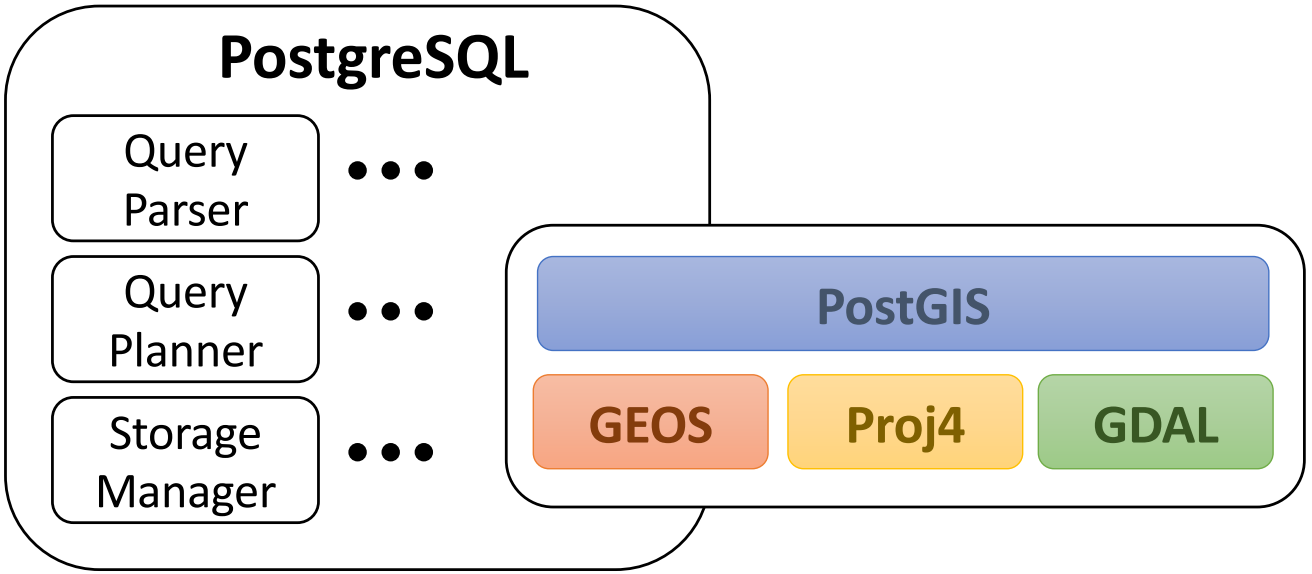
SGBD-OR: UDTs mais Complexos

```
CREATE TYPE Geometry
(
internallength = variable,
input = geometry_in,
output = geometry_out,
send = geometry_send,
receive = geometry_recv,
typmod_in = geometry_typmod_in,
typmod_out = geometry_typmod_out,
delimiter = ':',
alignment = double,
analyze = geometry_analyze,
storage = main);
```

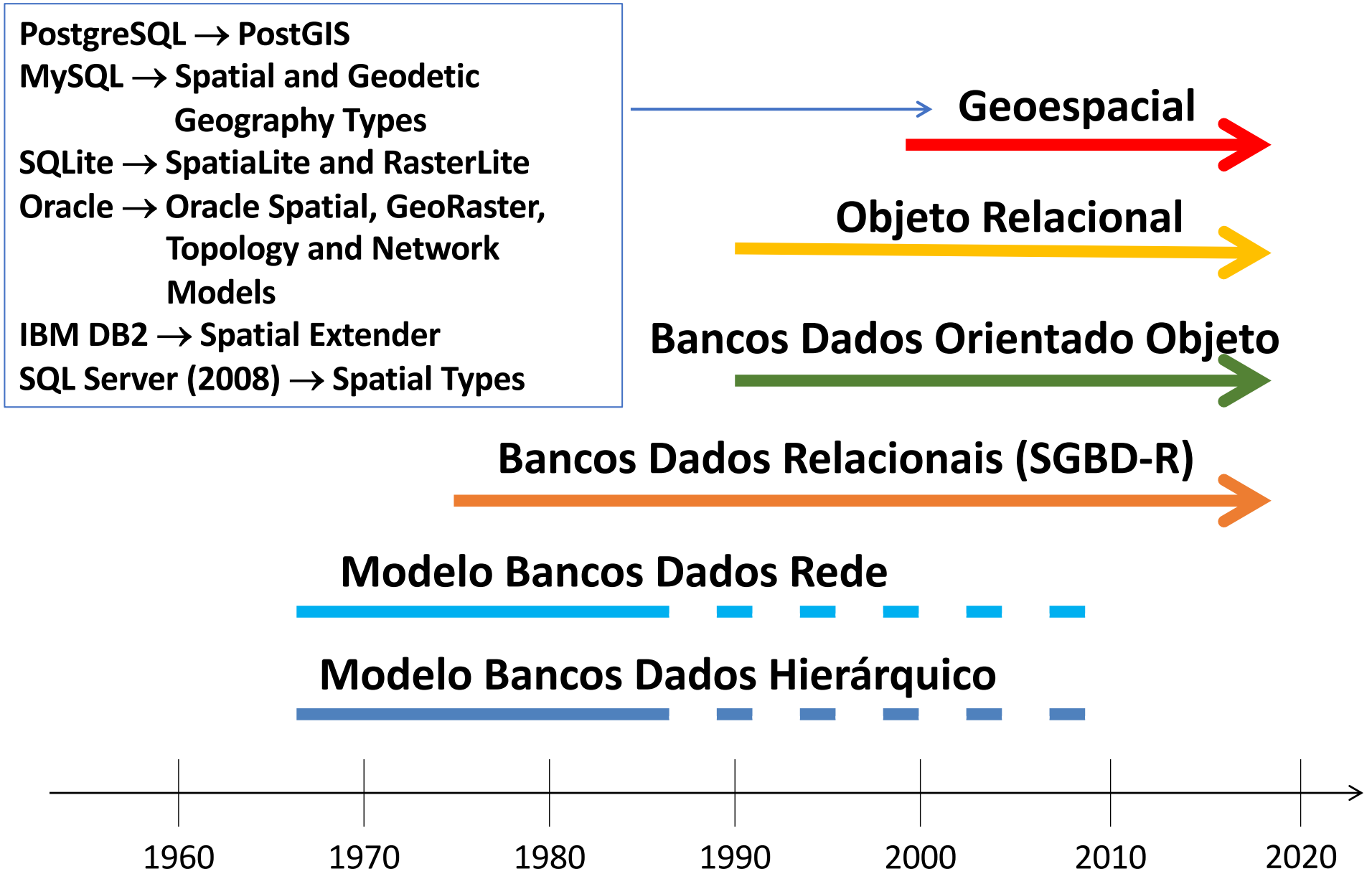
```
CREATE OR REPLACE FUNCTION _ST_Touches(geom1 geometry, geom2 geometry)
        RETURNS boolean
        AS '$libdir/postgis-2.1','touches'
        LANGUAGE 'c' IMMUTABLE STRICT
        COST 100;
...
```

PostGIS

- Projeto de software livre (GPLv2) desenvolvido inicialmente pela empresa Canadense Refractions Research: http://postgis.refractions.net
- Extensão geográfica para o SGBD-OR PostgreSQL:
  - Inicialmente:
    - Tipos geométricos e operadores espaciais OGC SFS
    - Índice espacial: árvore-R sobre GiST.
  - Atualmente:
    - Tipos circulares e compostos, 3D, Tipo geográfico, Raster, Topologia, Redes, Geocodificação de endereços

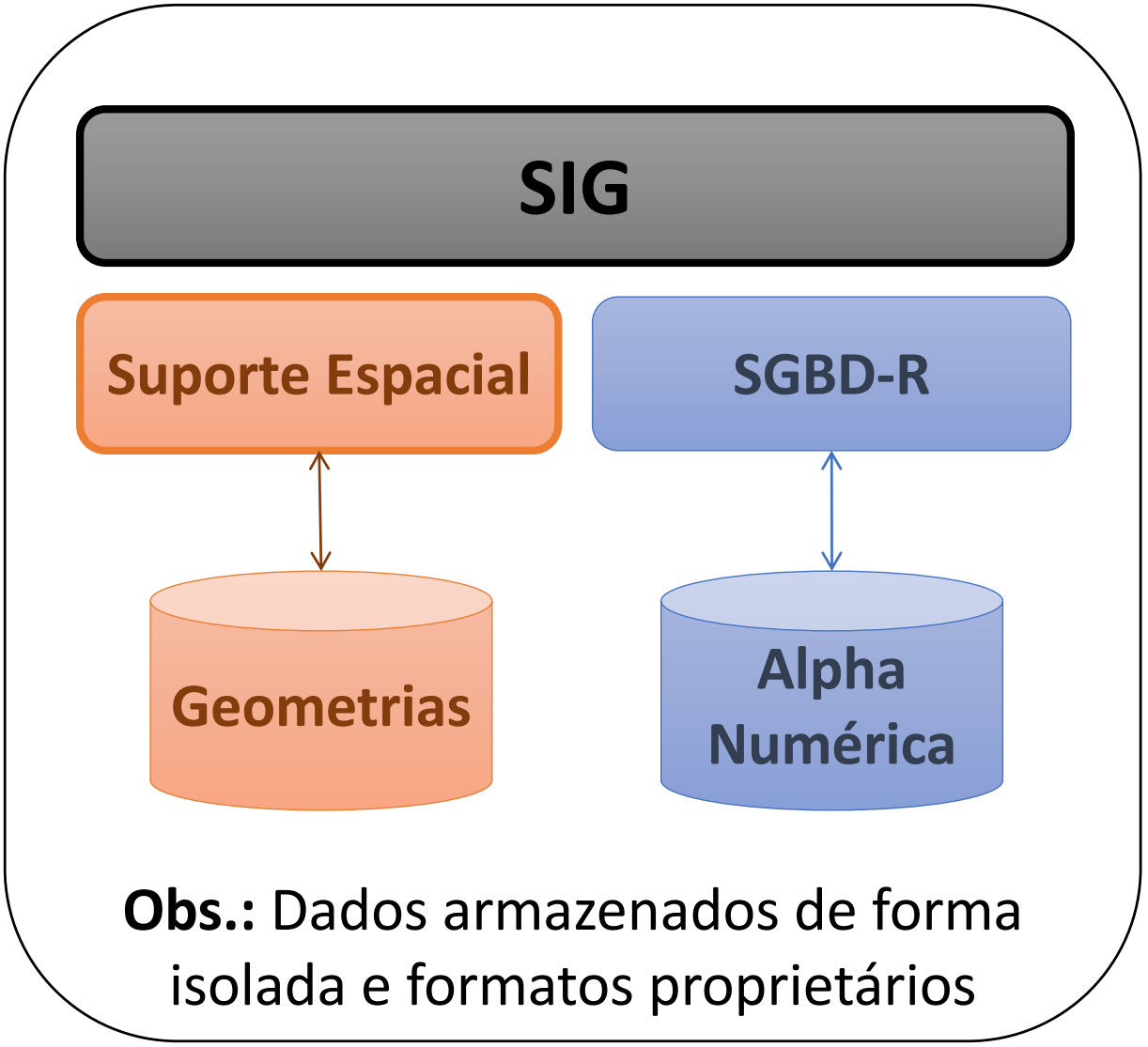


Evolução das Tecnologias de Bancos Dados



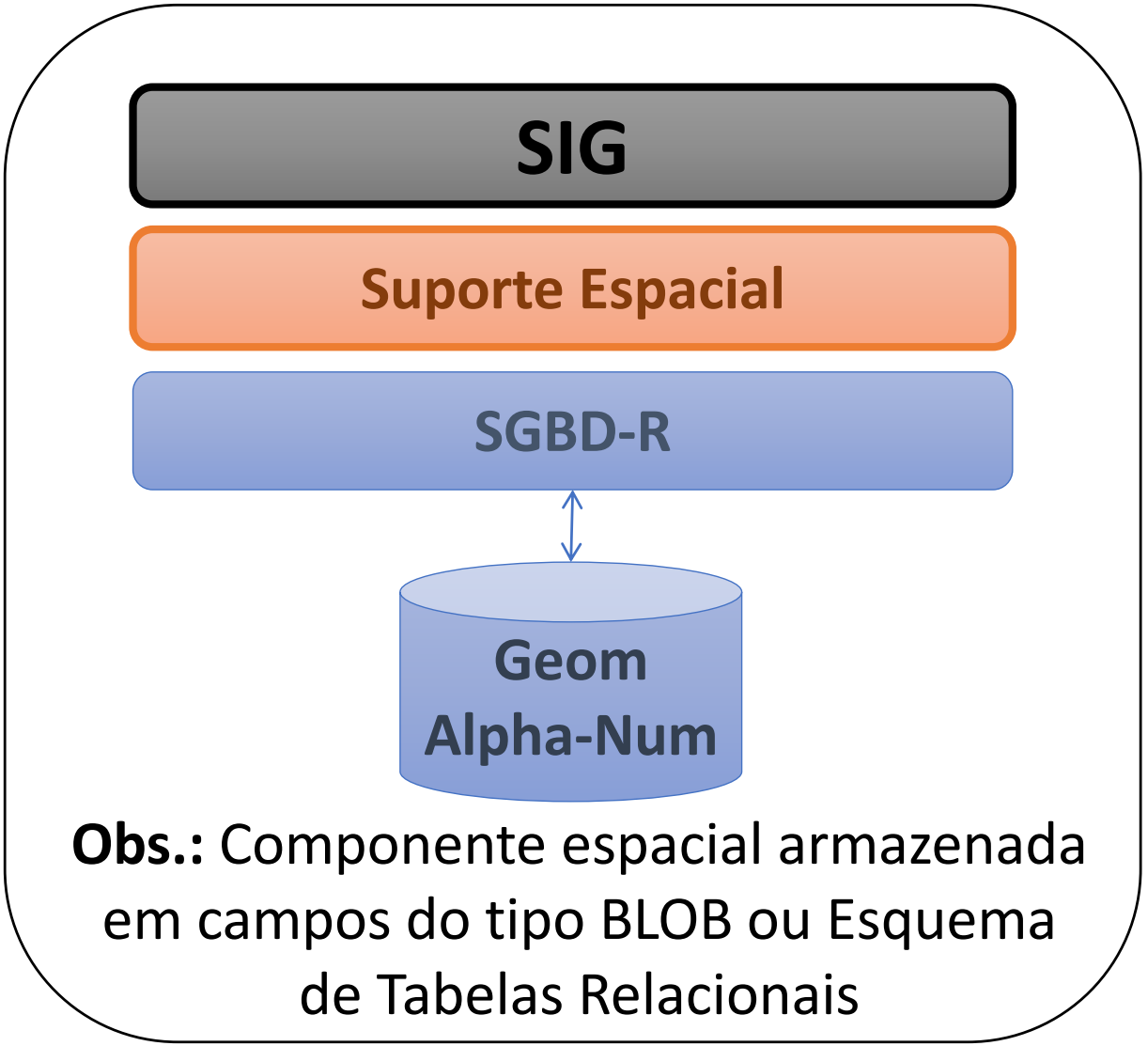
Antes dos anos 2000, como era a integração SIG e SGBD?

Arquitetura Dual



Obs.: Dados armazenados de forma isolada e formatos proprietários

Arquitetura em Camadas



Obs.: Componente espacial armazenada em campos do tipo BLOB ou Esquema de Tabelas Relacionais

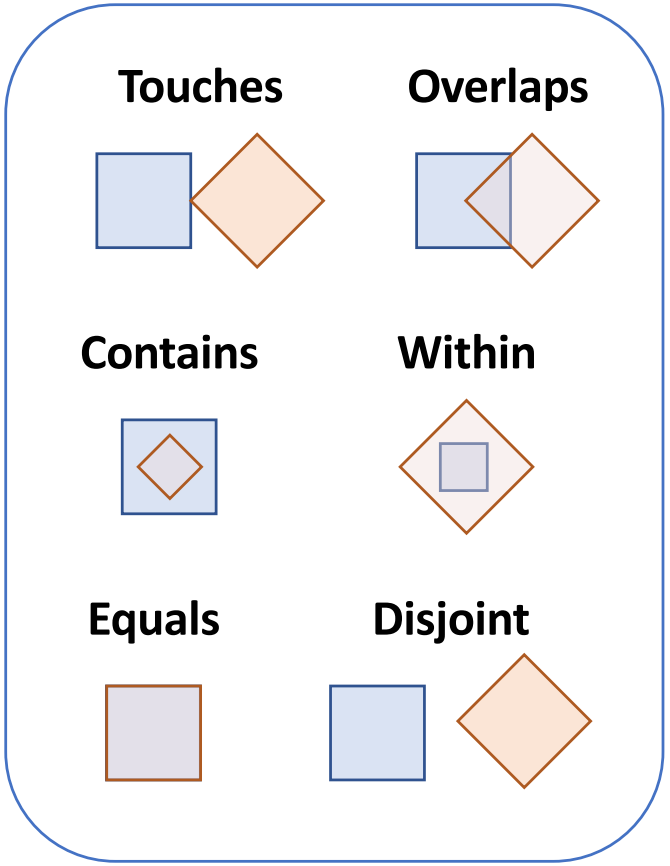
Com a integração do suporte espacial nos SGBD-R como passou a ser a integração SIG $\leftrightarrow$ SGBD-R?

Arquitetura Integrada: Tipos de Dados Geoespaciais

Tabela com Feições

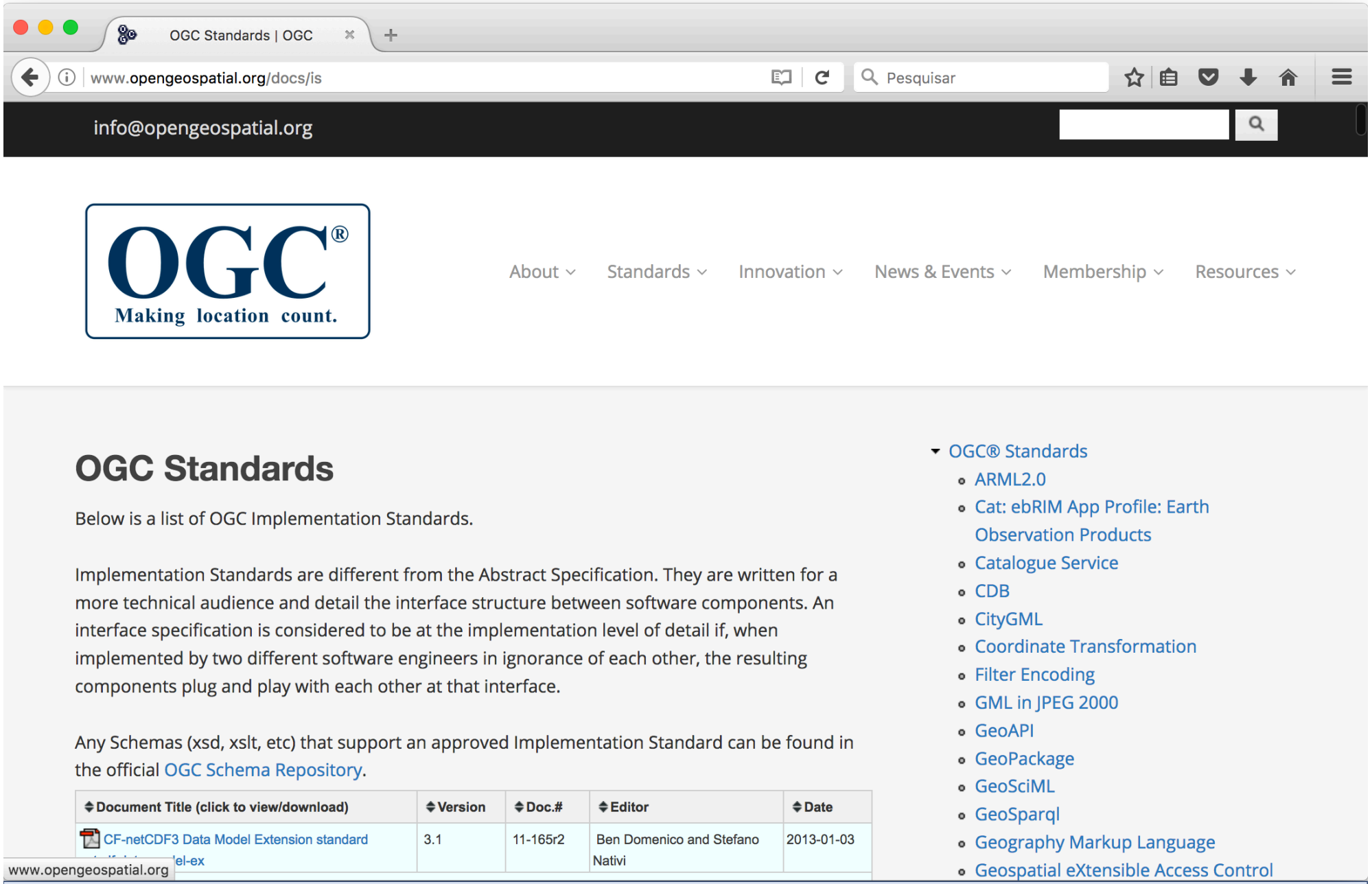
unidades_federativas				
ufid	**nome**	**populacao**	**e_vida**	fronteira
31	Minas Gerais	20.997.560	77	
35	São Paulo	44.396.484	77,8	
...	...	...	...	...

Operações Espaciais



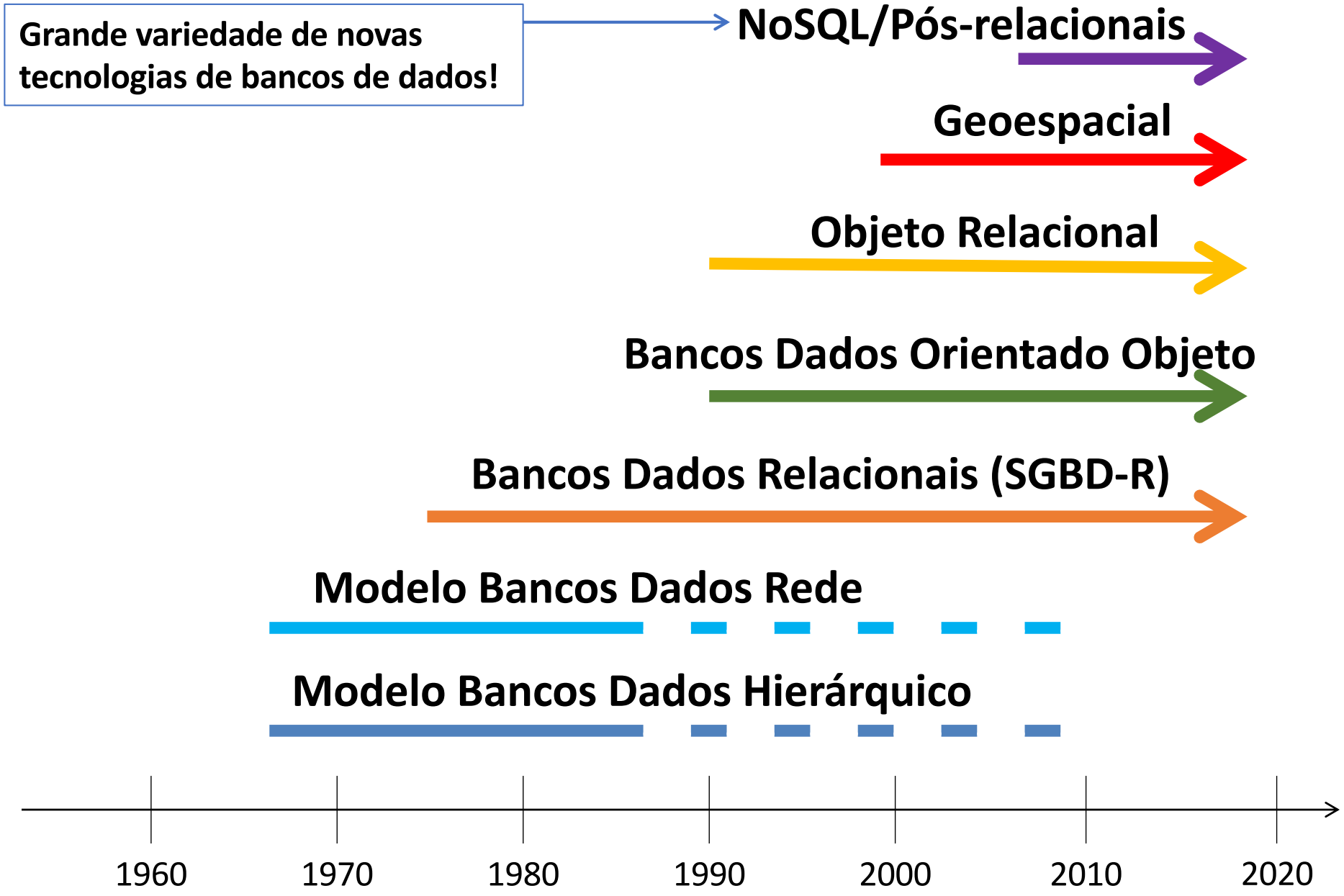
Obs.: Padrões OGC Simple Feature e ISO/SQL-MM Spatial

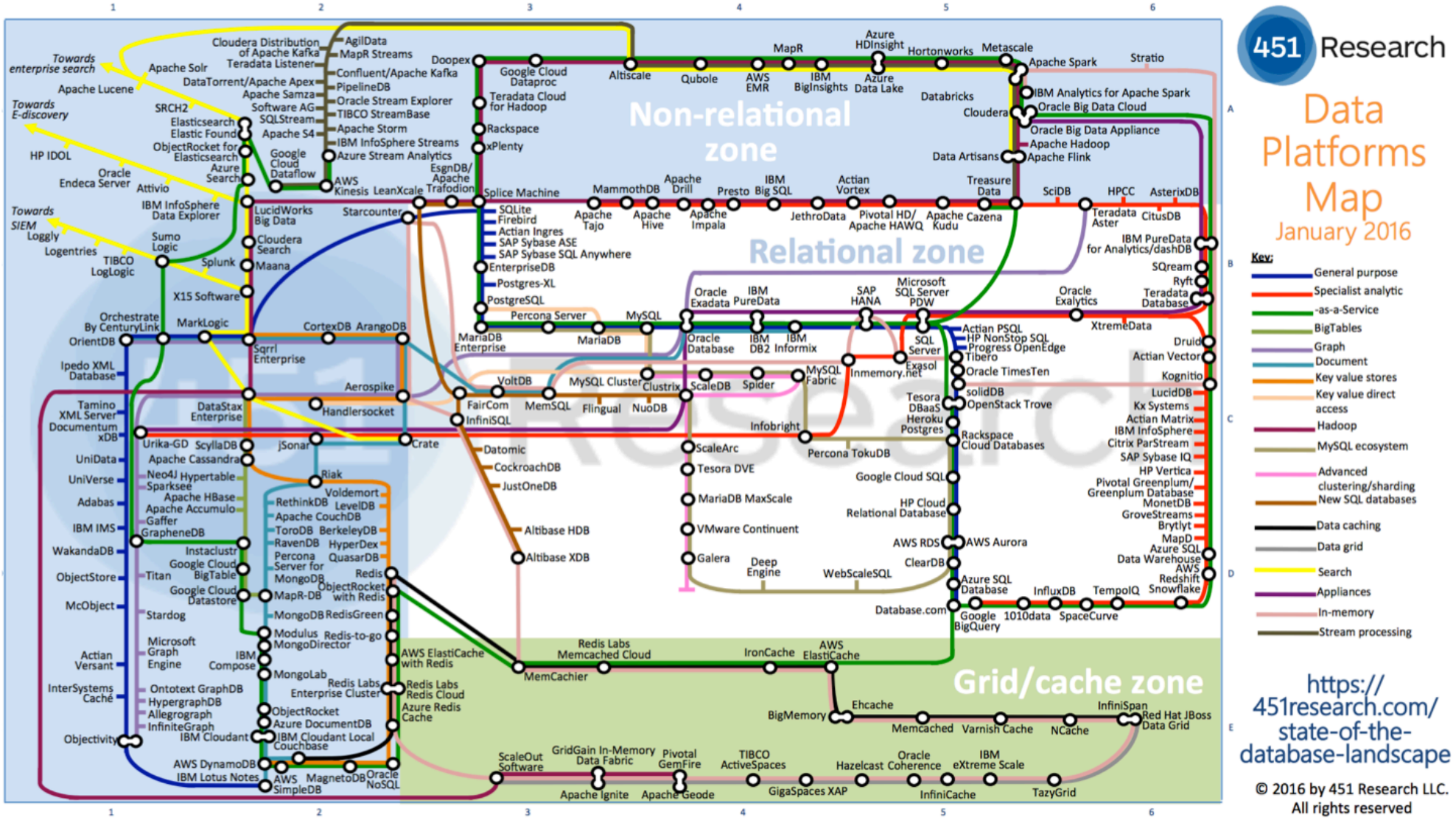
Padrões SQL:
OGC SFSQL e ISO SQL/MM Spatial



Open Geospatial Consortium

Evolução das Tecnologias de Bancos Dados



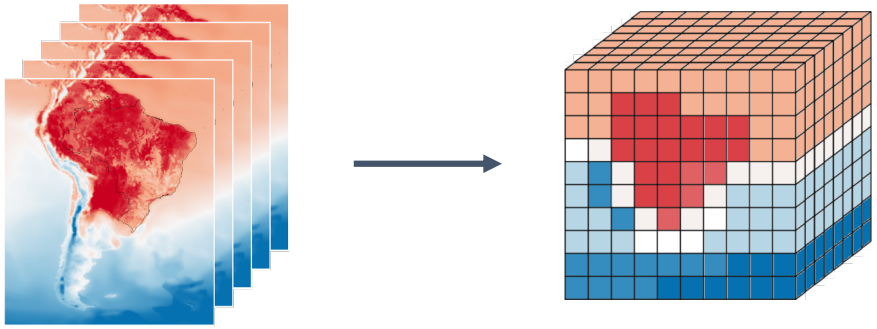


Tecnologias de Bancos de Dados

Sistemas de Bancos de Dados Matriciais

SGBD-M

- **Sistemas Gerenciadores de Banco de Dados Matriciais** (SGBD-M) – *Array Databases*
- Modelo de dados trata de "*Arrays*"



Modelo mais apropriado para dados intrinsecamente ordenados de forma espacial e/ou temporal

Array Databases

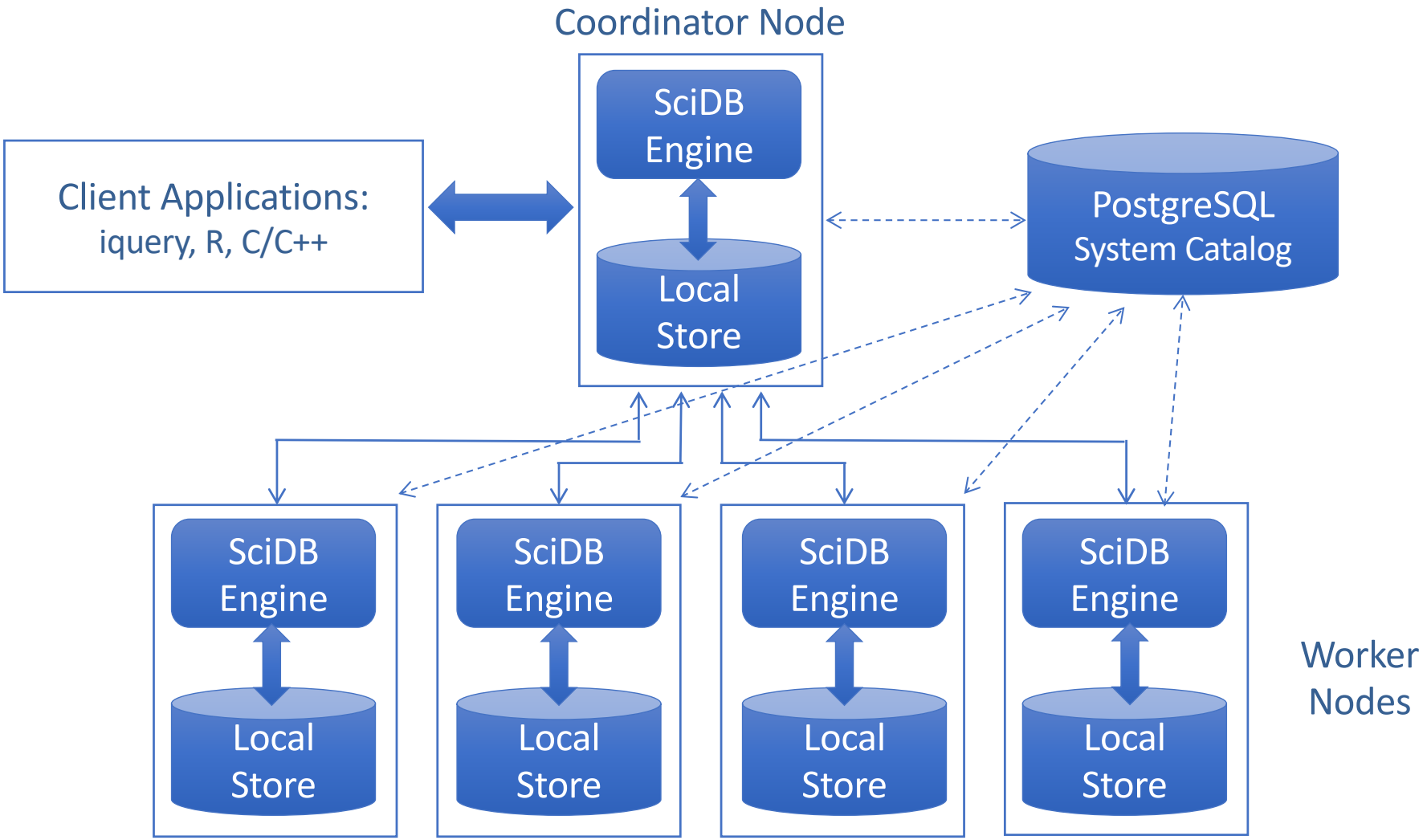
SciDB

Foto: David.Monniaux, CC BY-SA 4.0

"SciDB is an open-source analytical database oriented toward the data management needs of scientists."

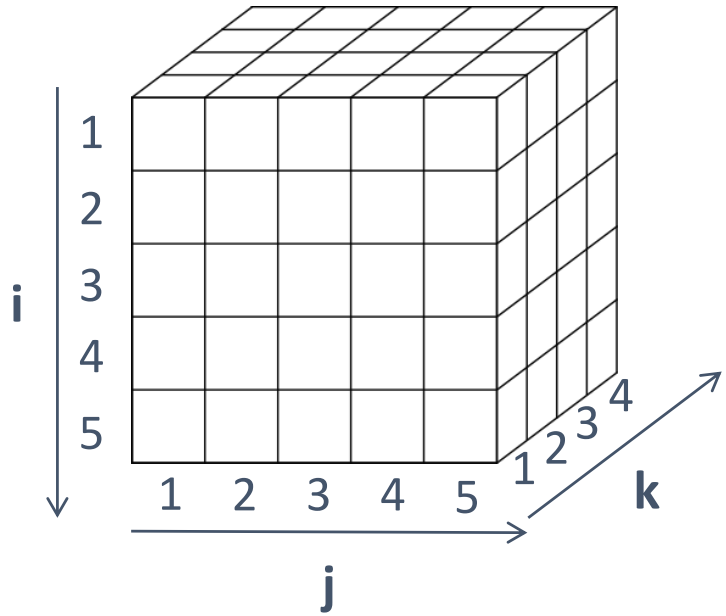
(Stonebraker et al., 2011)

SciDB: Arquitetura



Fote: Adaptado de Paradigm4 (2016)

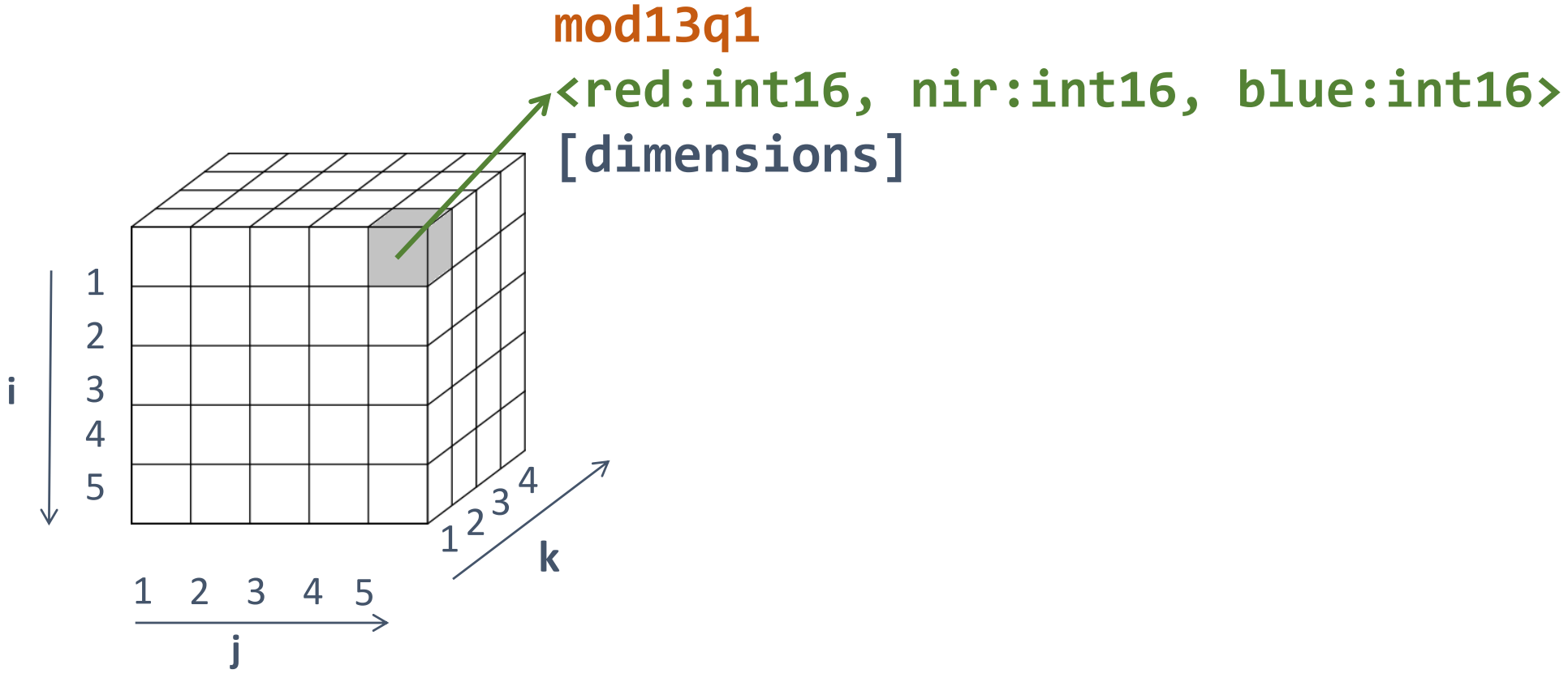
Arrays



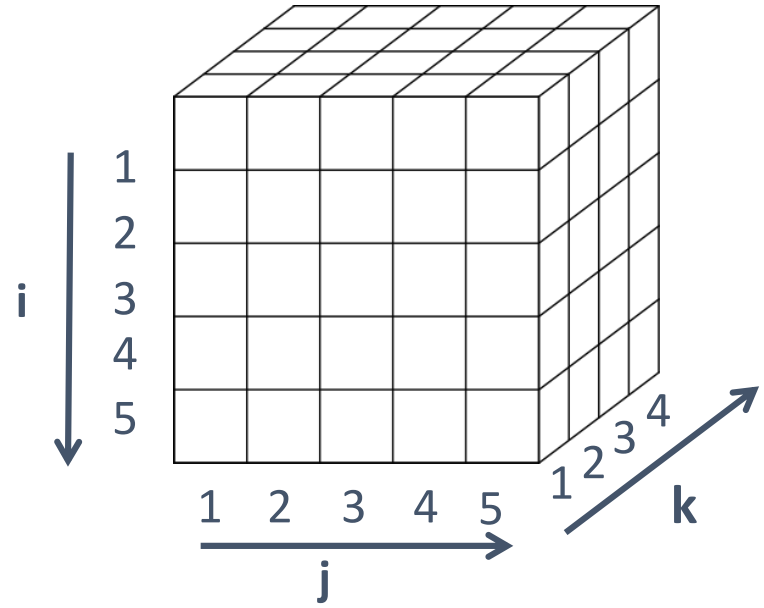
nome <atributos> [dimensões]

SciDB: Atributos

- Cada célula pode ser associada a múltiplos valores.
- Tipos de dados: `bool, char, datetime, datetimez, double, float, int8, int16, int32, int64, string, uint8, uint16, uint32, uint64`

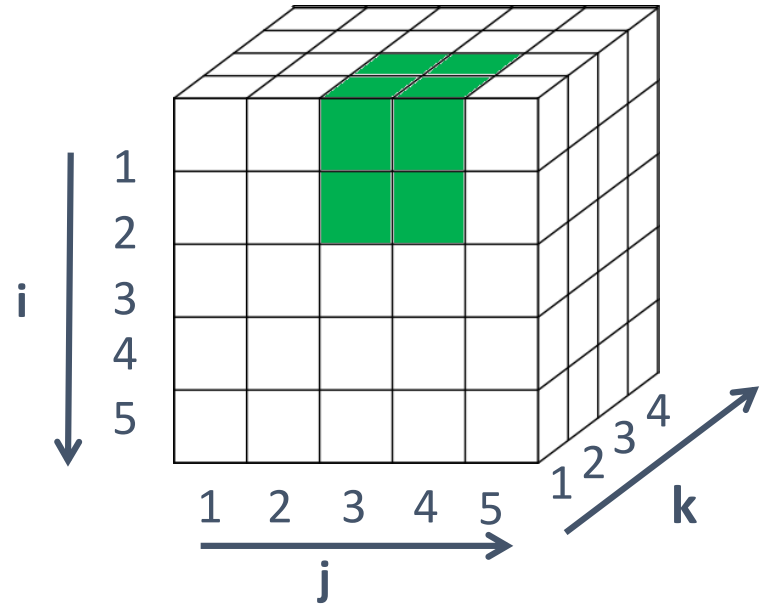


SciDB: Dimensões




```
mod13q1 <red:int16, nir:int16, blue:int16>
[j=1:5,2,1, i=1:5,2,1, k=1:4,2,1]
```

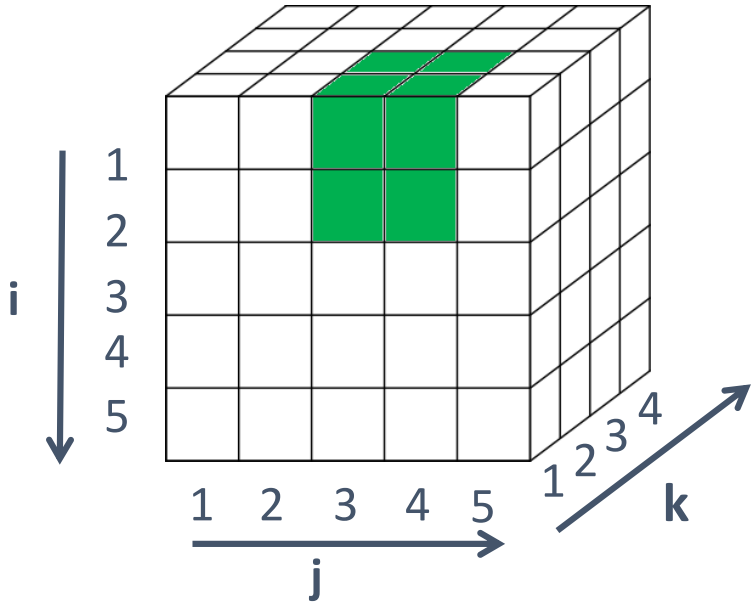
SciDB: Chunks

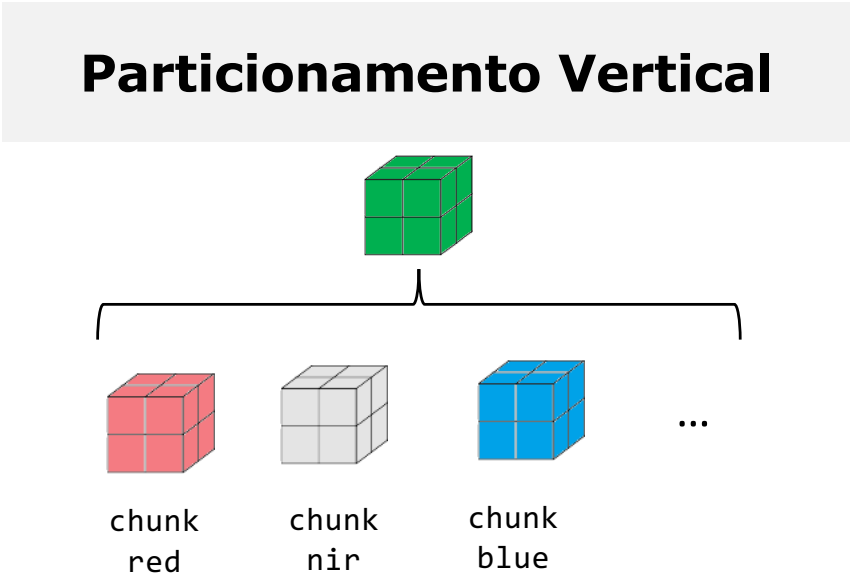



```
mod13q1 <red:int16, nir:int16, blue:int16>
[j=1:5,2,1, i=1:5,2,1, k=1:4,2,1]
```

Chunk size: **2** x **2** x **2**

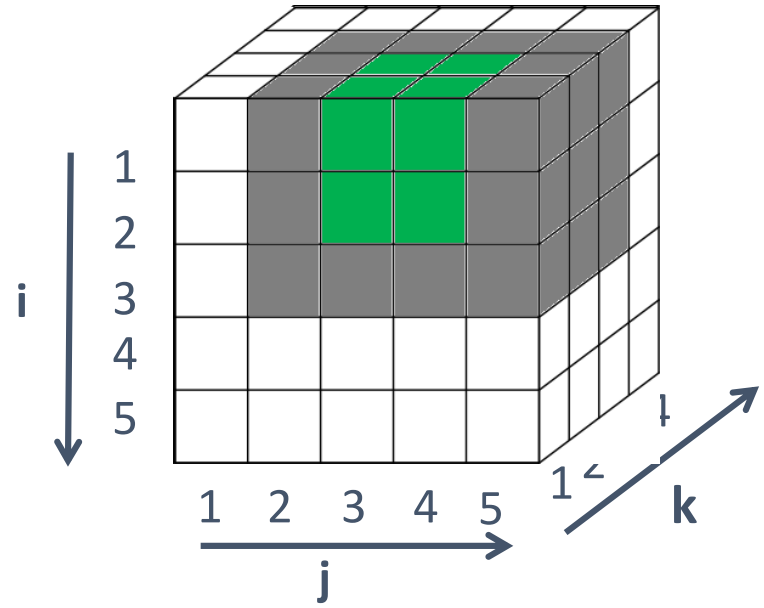
SciDB: Particionamento Vertical






```
mod13q1 <red:int16, nir:int16, blue:int16>
[j=1:5,2,1, i=1:5,2,1, k=1:4,2,1]
```

SciDB: Overlap (Replicação)




```
mod13q1 <red:int16, nir:int16, blue:int16>
[j=1:5,2,1, i=1:5,2,1, k=1:4,2,1]
```

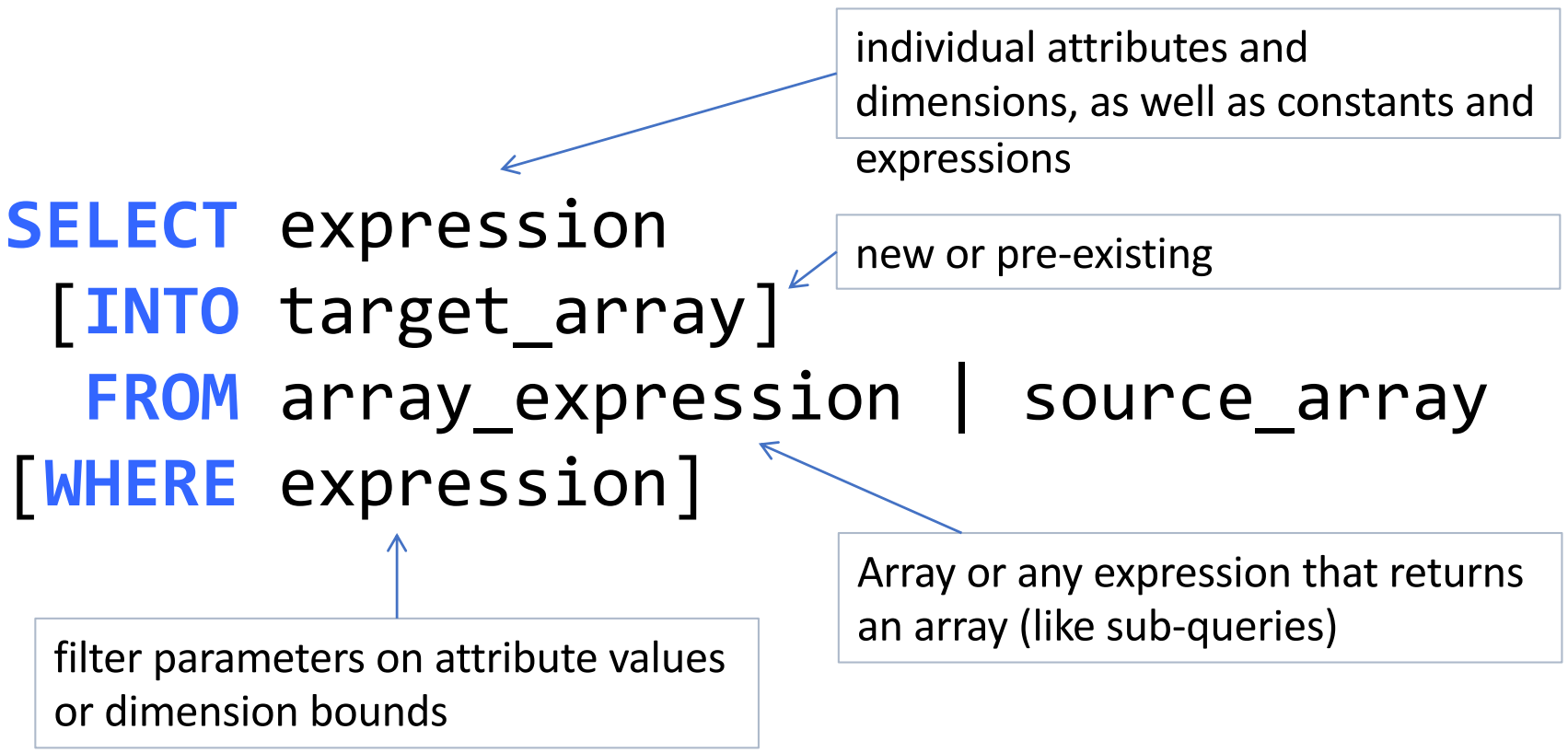
Chunk overlap: 1 x 1 x 1

Linguagens de Consulta

Array **Q**uery **L**anguage (AQL)

Array **F**unctional **L**anguage (AFL)

Array Query Language: AQL



There are DML and DDL clauses

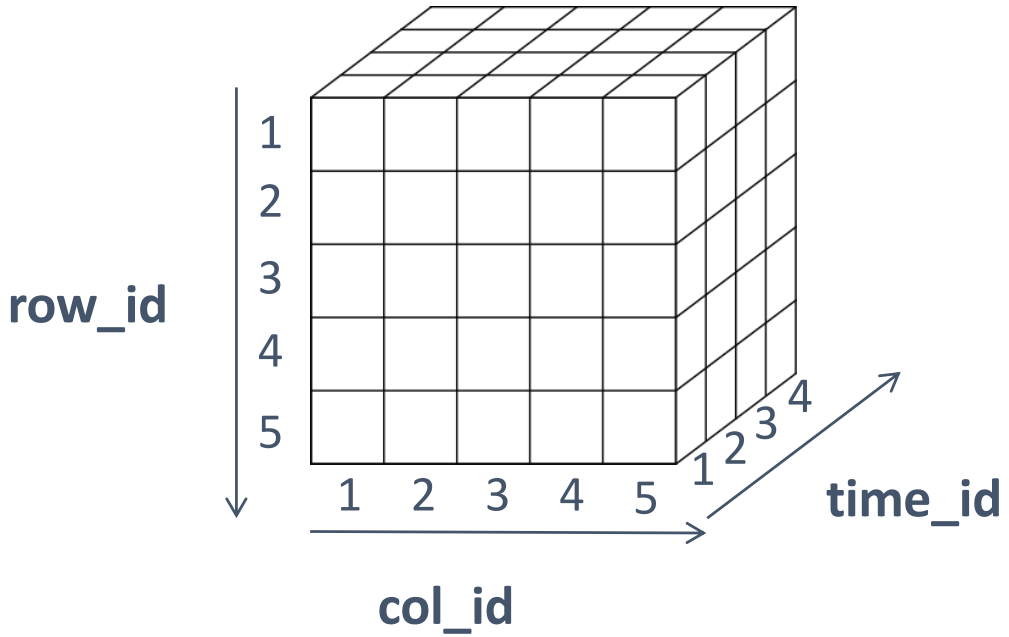
Array Functional Language (AFL)

```
store(
  build(<num:double>
        [x=0:8,1,0, y=0:9,1,0],
        random()
  ),
  random_numbers);
```

Consultas em AFL

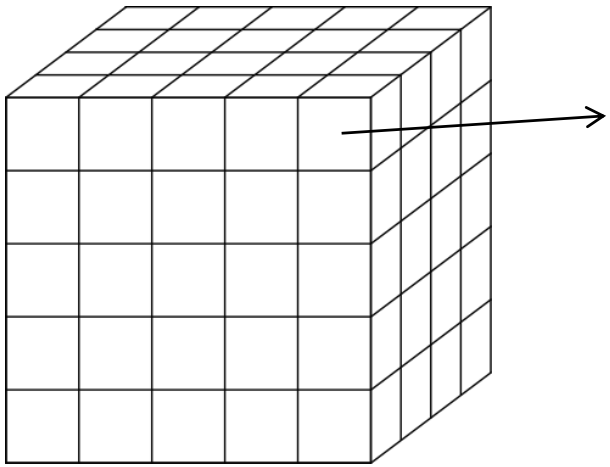
Definindo um Array: `mod13q1`

```
CREATE ARRAY mod13q1
<nir:double, red:double, blue:double>
[col_id=0:4,1,0, row_id=0:4,1,0,
 time_id=0:3,4,0]
```



Creating Array: mod13q1

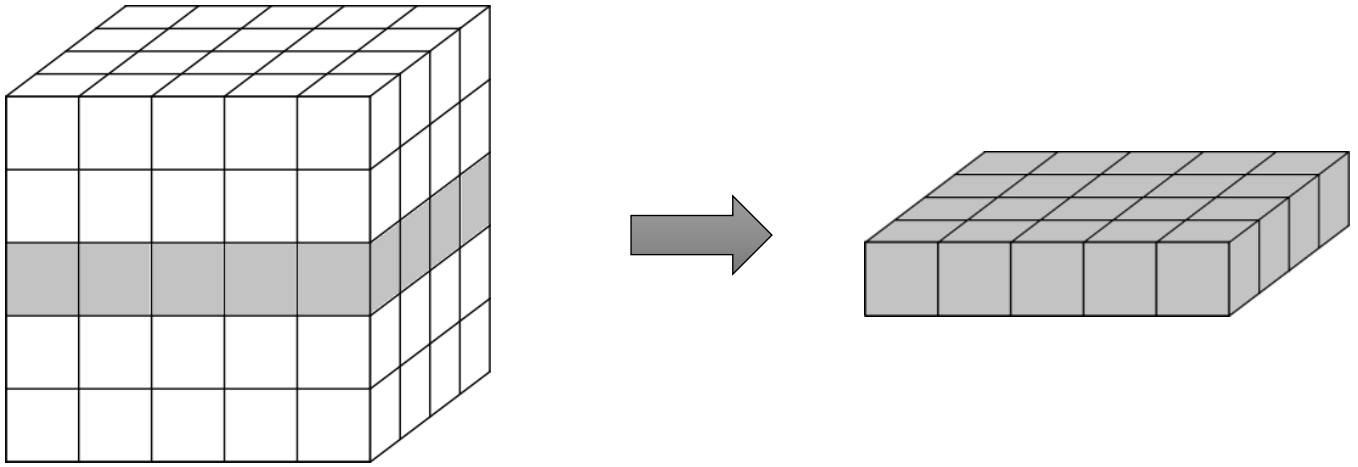
```
store(join(join(
build(<val:double>[col_id=0:4,1,0,row_id=0:4,1,0,time_id=0:3,4,0]
,(col_id+(row_id*5)+time_id*(5*5))/1.0),
build(<val:double>[col_id=0:4,1,0,row_id=0:4,1,0,time_id=0:3,4,0]
,(col_id+(row_id*5)+time_id*(5*5))/10.0)),
build(<val:double>[col_id=0:4,1,0,row_id=0:4,1,0,time_id=0:3,4,0]
,(col_id+(row_id*5)+time_id*(5*5))/100.0) ), mod13q1)
```



```
nir  = val/1.0
red  = val/10.0
blue = val/100.0
```

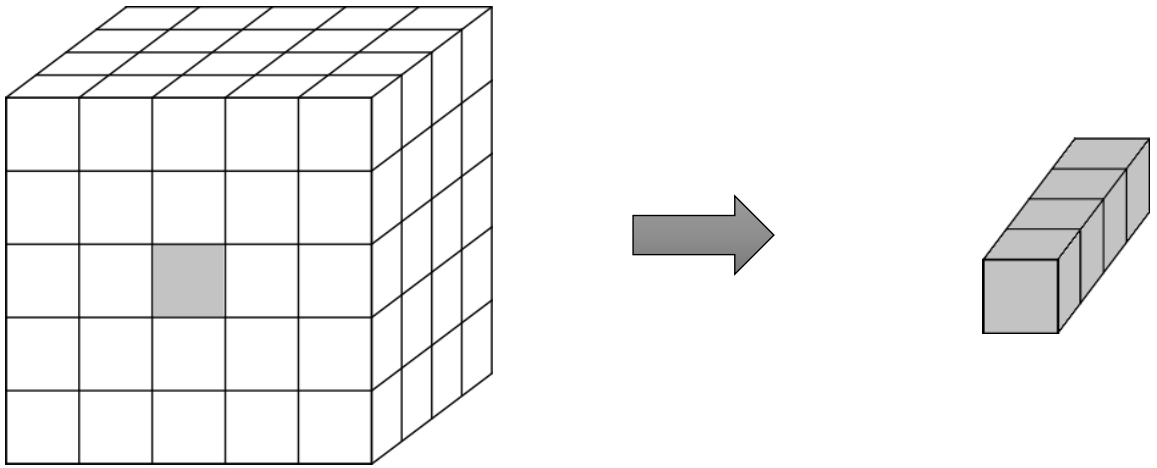
Horizontal Slice

```
subarray(mod13q1,
         0, 2, 0,
         4, 2, 3)
```



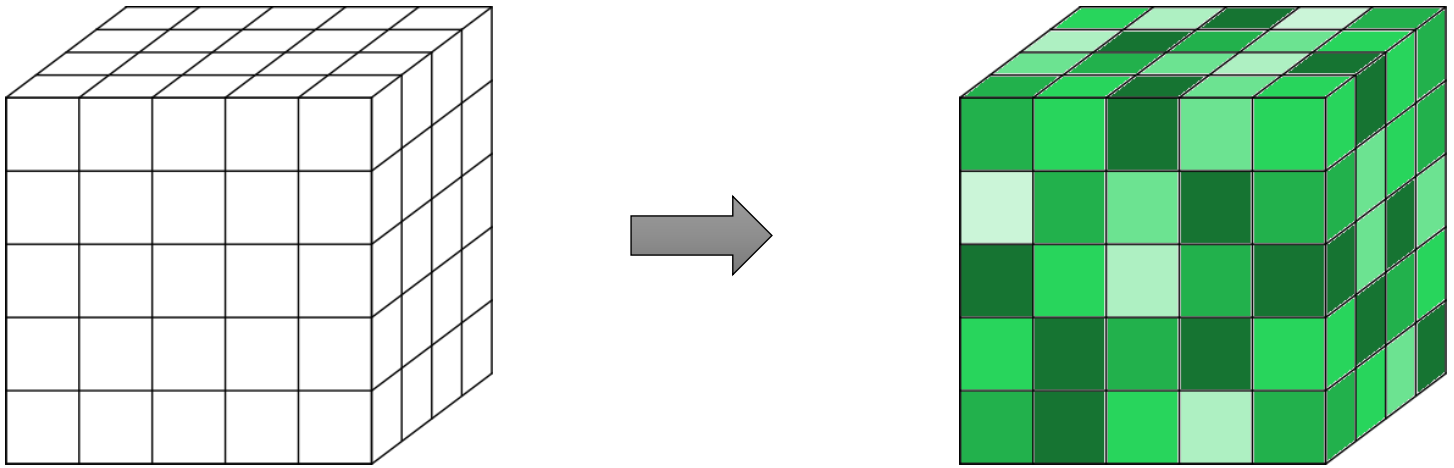
Time Series

```
subarray(mod13q1,
         2, 2, 0,
         2, 2, 3)
```



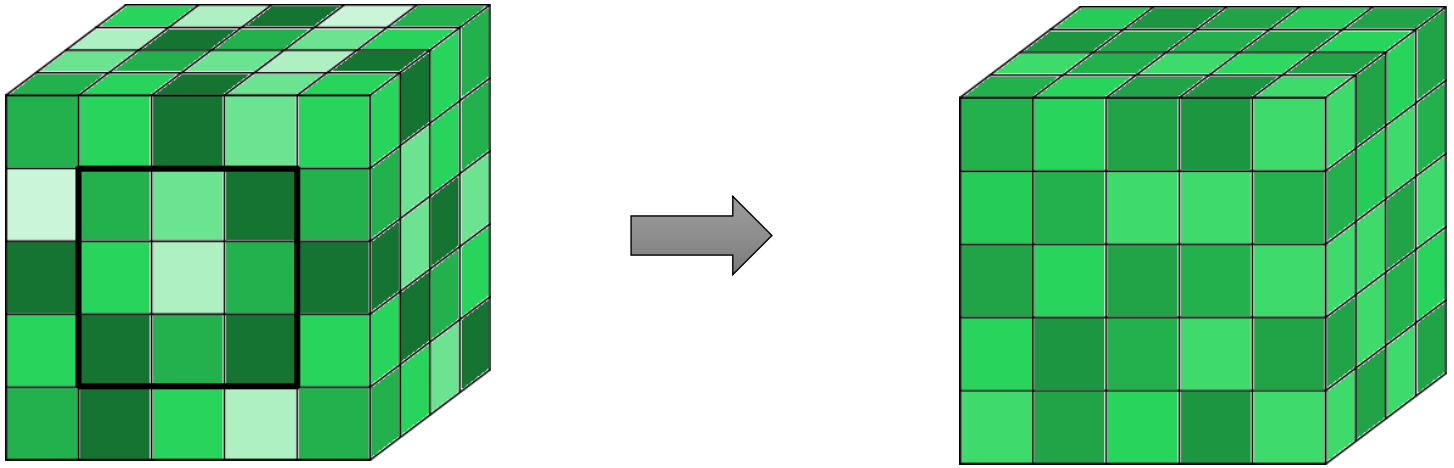
NDVI

```
store (
  project (
    apply (
      mod13q1, new_evi,
      2.5*(nir-red) /(nir+6.0*red-7.5*blue+1.)
    ),
    new_evi ), evi_array);
```



Window Queries

```
store(
    window(evi_array,
           0, 2,
           0, 2,
           0, 0,
           avg(new_evi)
    ),
   evi_avg);
```

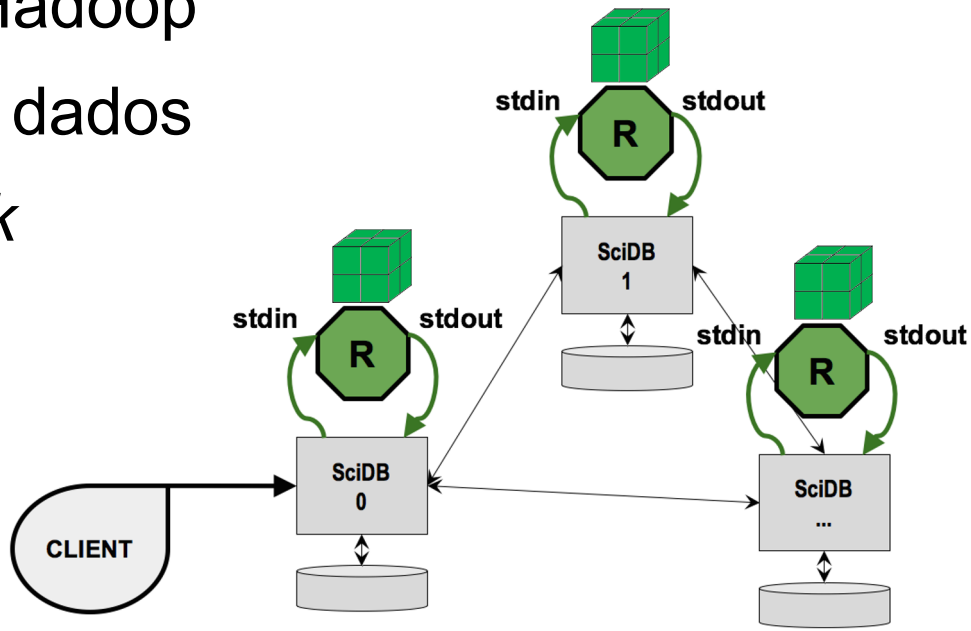


Stream

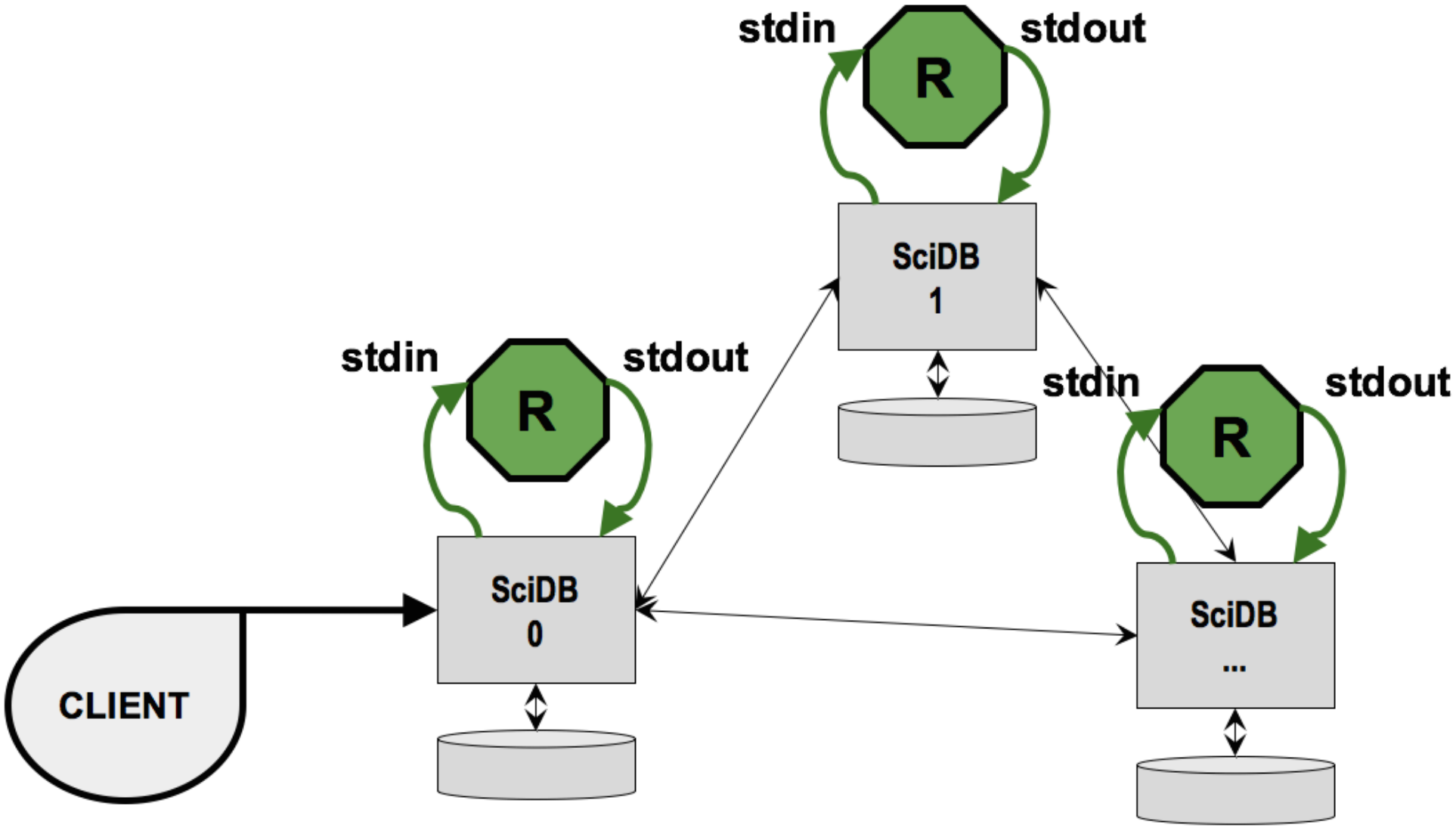
SciDB: Stream

- Plugin Stream
- Abordagem equivalente Hadoop
- Processamento junto aos dados
- Processamento por *chunk*

```
stream(array,'/path/to/app');
```



Fonte: Paradigm4 (2018)

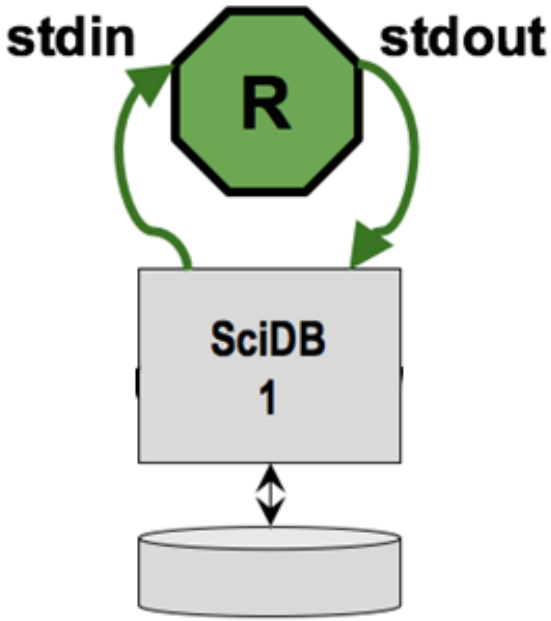


SciDB – Stream

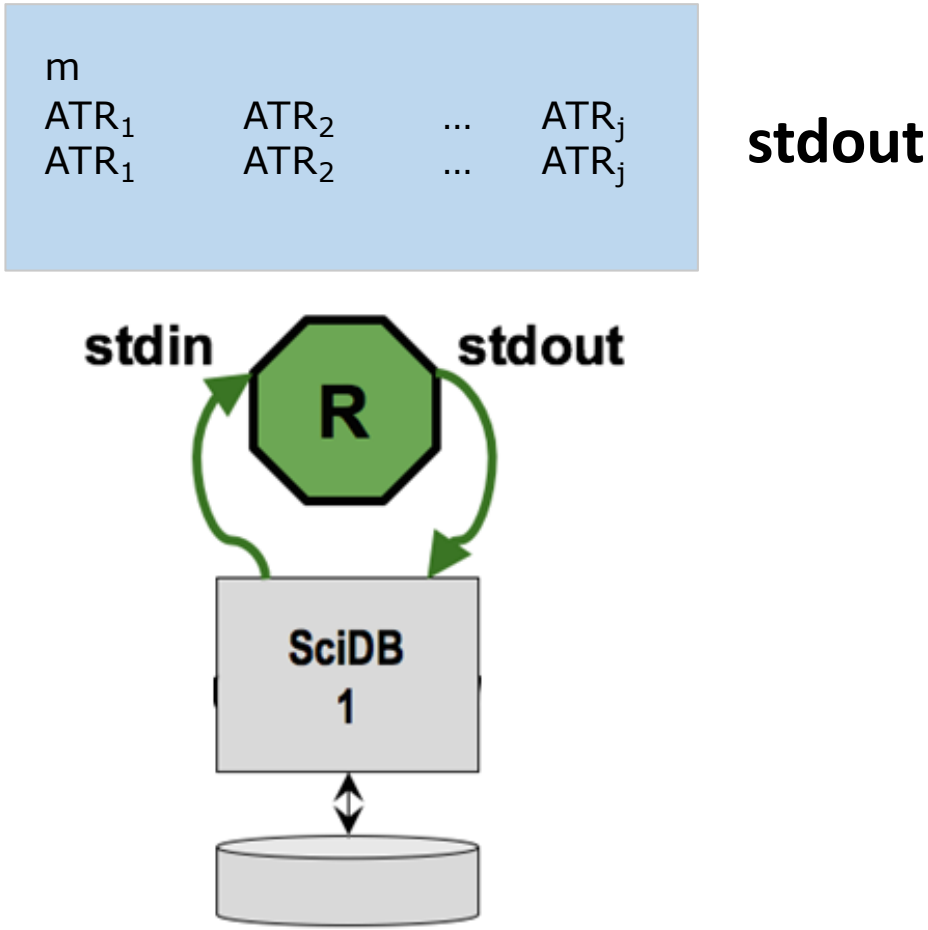
SciDB -> Application

stdin

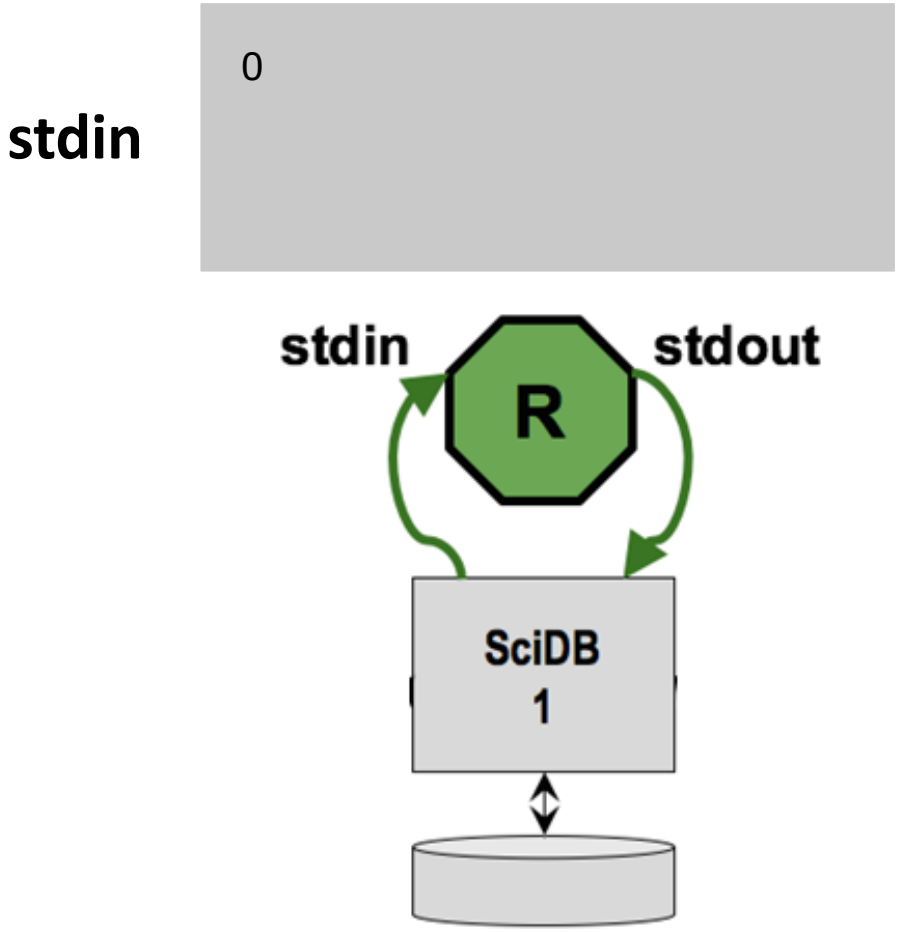
n			
ATR_1	ATR_2	...	ATR_k
ATR_1	ATR_2	...	ATR_k
ATR_1	ATR_2	...	ATR_k



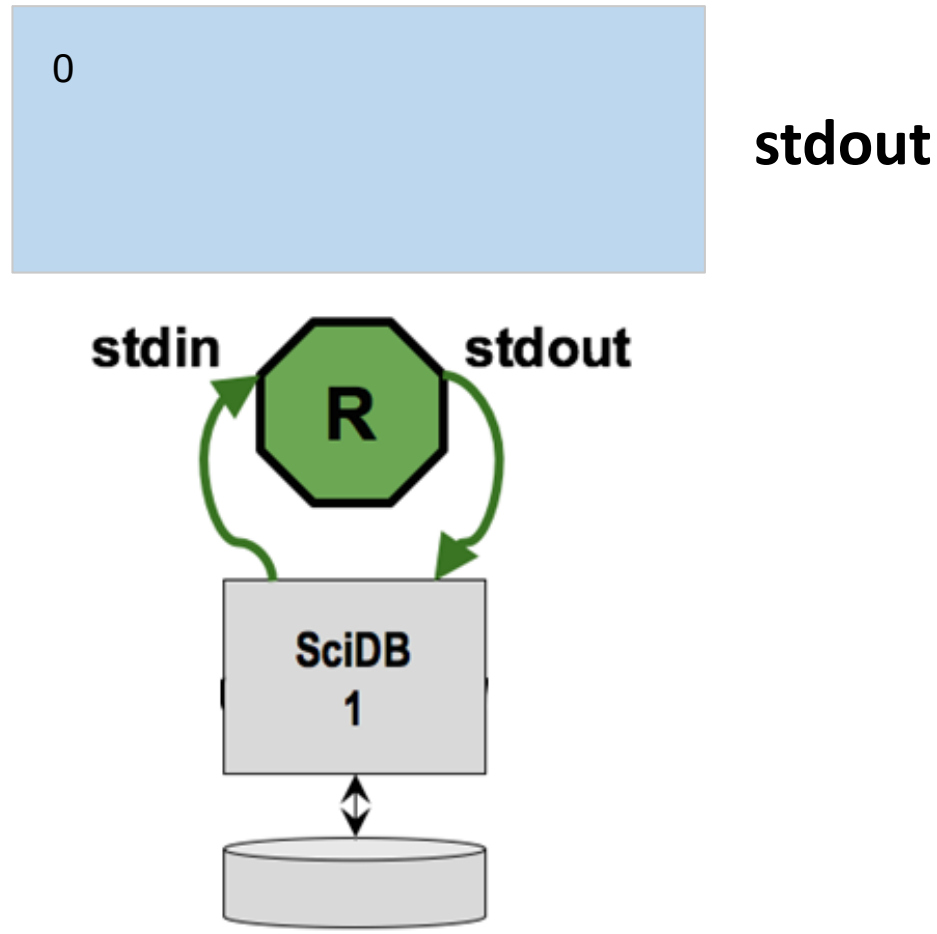
Application -> SciDB



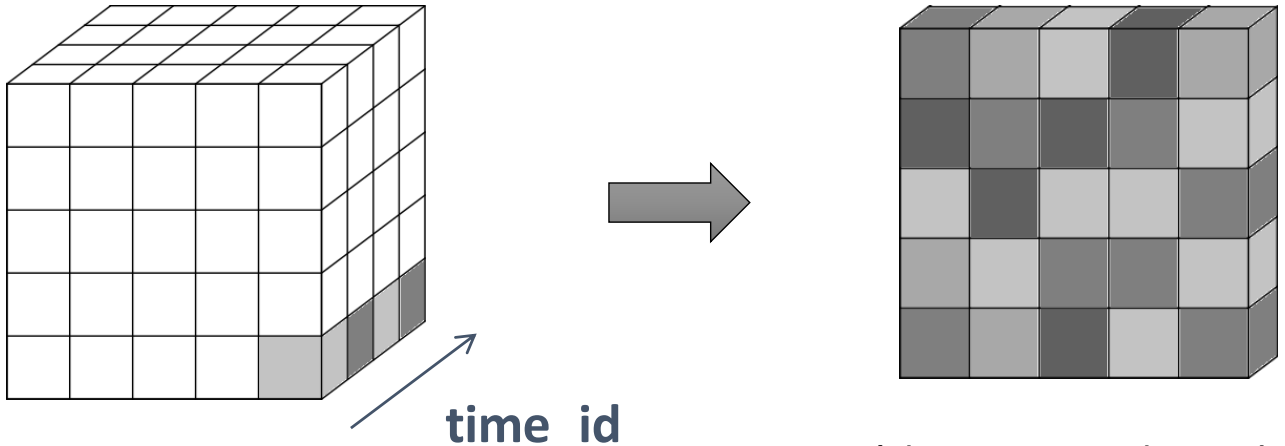
SciDB: Finished



Application: Finished



Computing the Average of Time Series



Médias para cada atributo

```
stream(mod13q1, '/GeoData/aqua/scidb/avg_time.py')
```

Considerações Finais

Considerações Finais

- Nesta aula apresentamos um panorama geral das tecnologias de bancos dados, com foco na tecnologia relacional.

- Discutimos a criação de novos tipos de dados e operações.

- Também foi apresentado, de maneira breve, um exemplo de sistema de bancos de dados baseado em um modelo orientado a arrays: SciDB.

Referências Bibliográficas

Referências Bibliográficas

- Stonebraker, M.; Brown, P.; Poliakov, A.; Raman, S. **The Architecture of SciDB**. Proceedings of the 23rd International Conference on Scientific and Statistical Database Management, SSDBM'11, 2011.

- Paradigm4. SciDB Documentation. Acesso: Agosto de 2016.

Padrões e Especificações

- OGC. ***OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture***. Disponível em: http://www.opengeospatial.org. Acesso: Outubro de 2012.

- OGC. ***OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option***. Disponível em: http://www.opengeospatial.org. Acesso: Outubro de 2012.

- ISO. ***SQL Multimedia and Application Packages – Part 3: Spatial***.

Artigos

- R. H. Güting. ***Gral: an extensible relational database system for geometric applications***. In Proceedings of the 15th international conference on Very large data bases (VLDB '89). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 33-44.

- Vijlbrief, T., and P. van Oosterom. ***The GEO++ System: An Extensible GIS***. Proc. 5th Intl. Symposium on Spatial Data Handling, Charleston, South Carolina, 1992, pp. 40-50.

- Dangermond, J. ***A Classification of Software Components Commonly Used in Geographic Information Systems***. In Proceedings of the U.S.-Australia Workshop on the Design and Implementation of Computer-Based Geographic Information Systems, Honolulu, HI, 1982, pp. 70–91.

Artigos

- E. F. Codd. 1970. ***A relational model of data for large shared data banks***. *Communications of the ACM, v.* 13, n. 6, June 1970, pp. 377-387.

- Chen, P. ***The Entity-Relationship Model-Toward a Unified View of Data***. ACM Transactions on Database Systems, v. 1, n. 1, 1976, pp. 9-36.

- GRAY, J. ***Evolution of Data Management***. IEEE Computer, v. 29, n. 10, 1996, pp. 38-46.