physical design

unit 6

What is?

Optimization

Desnormalization

lsi Departamento de Lenguajes y Sistemas Informáticos
Universitat d'Alacant
Universidad de Alicante

design phases

Conceptual design

- Independent of DBMS
- Independent of the Data Model

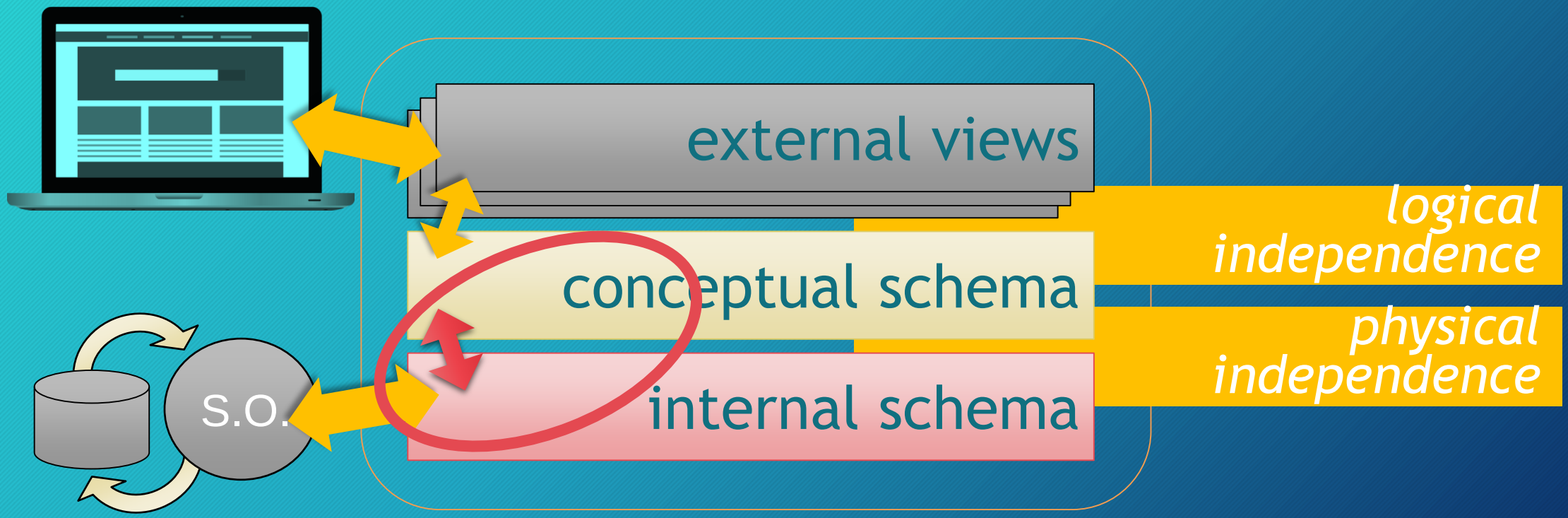
Logic design

- Independent of DBMS
- Dependent of the Data Model

Physical design

- Depends on DBMS

physical design?



transaction

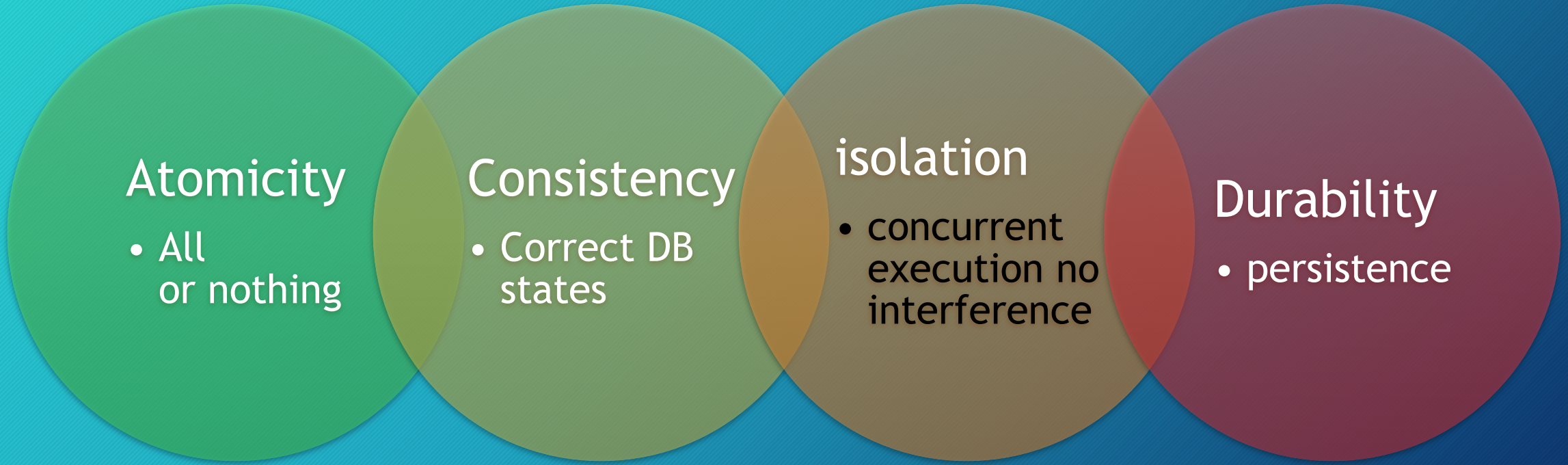
process or executable program

- n access to BD (read or write)
  - all (COMMIT) or nothing (ROLLBACK) is executed
- correct initial and final DB states

```
BEGIN TRANSACTION
    set x = (select avg(pvp) from
    artículo);
    update articulo set pvp = pvp+x/2;
    update linped set
    importe=importe+x;
    if (@@error) then ROLLBACK
COMMIT
```

transactions

BD transactional systems: ACID properties



physical design: objectives

Response time

transaction time from launch to response

Use of space

storage space files and structures access

Transaction productivity

average number of transactions per minute(peak system conditions)

physical design: inputs


```
marca varchar(15),
empresa varchar(60),
logo blob )
CP (marca)

articulo (
cod varchar(7),
nombre varchar(45),
pvp decimal(7,2),
marca varchar(15),
imagen blob,
urlimagen varchar(100),
especificaciones text)
CP (cod)
CAj (marca) -> marca

camara (
cod varchar(7),
resolucion varchar(15),
sensor varchar(45),
tipo varchar(45),
factor varchar(10),
objetivo varchar(15),
pantalla varchar(20),
zoom varchar(40))
CP (cod)
CAj (cod) -> articulo
```

fundamentos de las bases de datos

- Logical design output
  - set of relationships
- Queries, transactions and applications to be executed
  - Understand the workload of queries and updates
- User performance requirements
  - speed certain queries or updates
  - number of transactions per second to be processed

Physical design: tasks

- Transferring the Logical Data Model to the DBMS
  - creation of tables, restrictions, delete/update policies, stored procedure...
- Storage structures
  - storage types (files) and growth estimates
- Access roads (indexes)
- Denormalization
  - relax logical design
- Monitoring and Adjustments
  - monitor performance statistics, modify design

Physical design: outputs

- DB schema created in DBMS
  - there may be alternatives

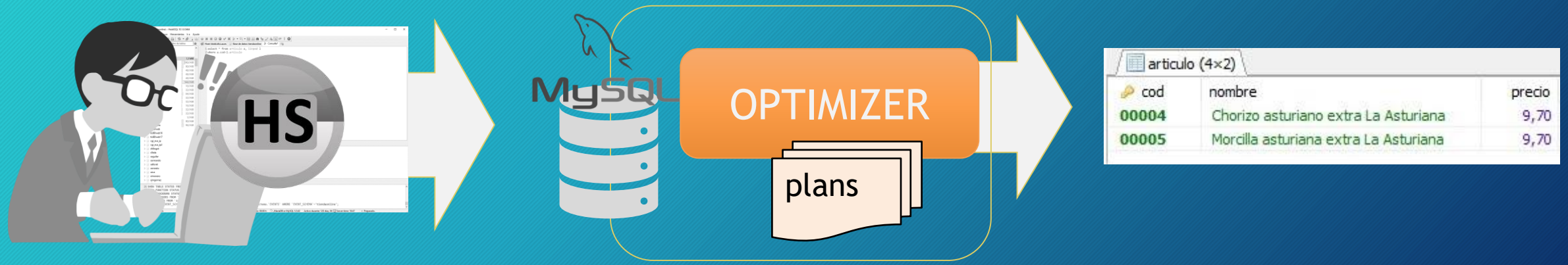
```
DROP TABLE IF EXISTS `asignado`; CREATE TABLE IF NOT EXISTS `asignado` (
  `idTrabajador` varchar(9) NOT NULL DEFAULT '',  `idTurno` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`idTrabajador`,`idTurno`), KEY `idTurno` (`idTurno`),   CONSTRAINT `asignado_ibfk_1` FOREIGN KEY (`idTrabajador`) REFERENCES `trabajador` (`colegiado`) ON DELETE CASCADE,   CONSTRAINT `asignado_ibfk_2` FOREIGN KEY (`idTurno`) REFERENCES `turno` (`id`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `diagnostico`; CREATE TABLE IF NOT EXISTS `diagnostico` (  `idH` int(10) NOT NULL DEFAULT '0',  `idL` int(10) NOT NULL DEFAULT '0',   `descripcion` varchar(200) DEFAULT NULL,   PRIMARY KEY (`idH`,`idL`),   KEY `fk_diagnosticolineahistorial` (`idL`,`idH`),  CONSTRAINT `fk_diagnosticolineahistorial` FOREIGN KEY (`idL`, `idH`) REFERENCES `linea_historial` (`id`, `idHistorial`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

indexes

Query Optimizer

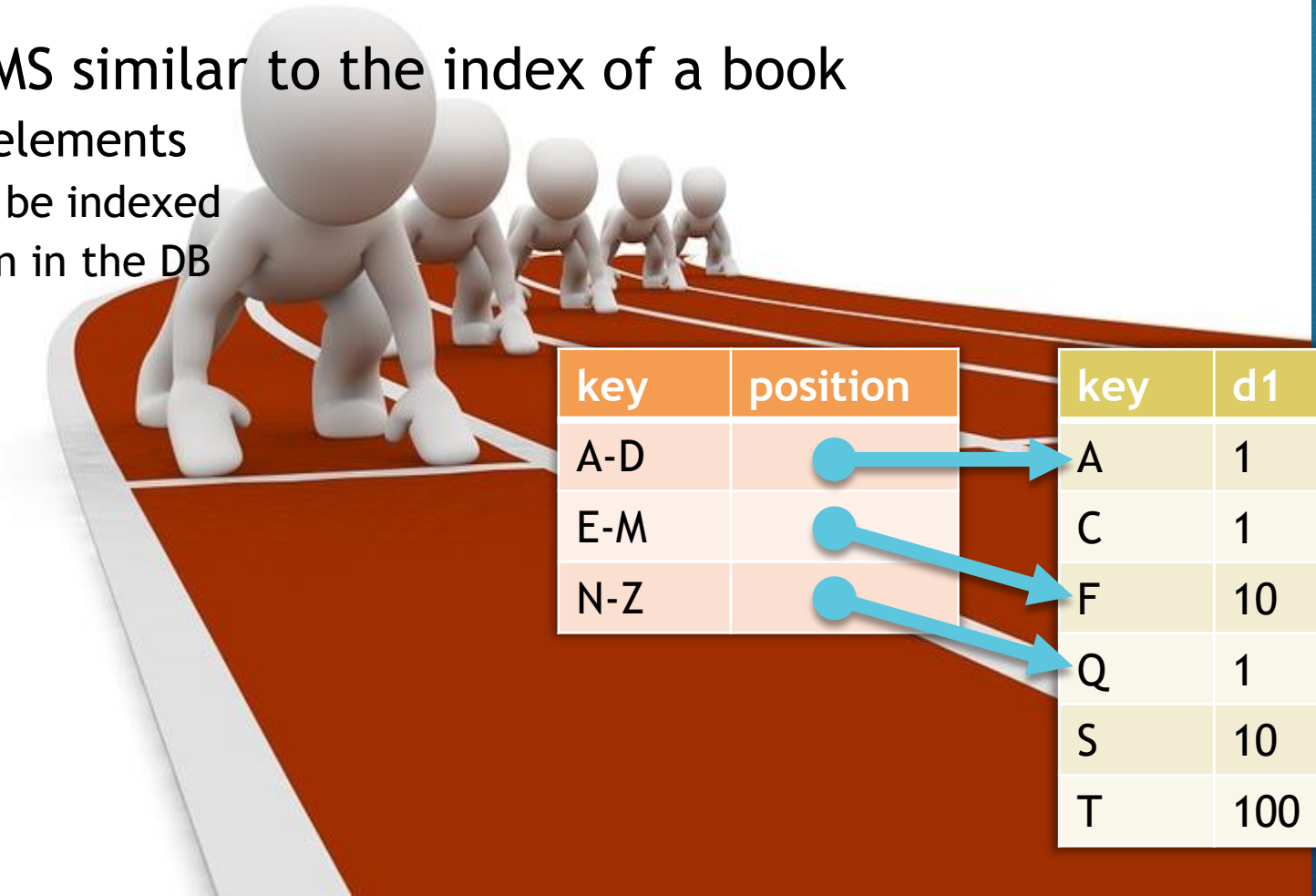
- PRE-EXECUTION PLAN
  - Select an execution plan from among the possible ones to solve the query
  - It's supposed to be optimal but
- The optimum plan is not always selected



indexes

An index in a DBMS similar to the index of a book

- Keep pairs of elements
  - Element to be indexed
  - The position in the DB



key	position
A-D	
E-M	
N-Z	

key	d1	d2	d3
A	1	safaf	ssss
C	1	dddd	ssss
F	10	null	null
Q	1	dsfa	null
S	10	null	sadfa
T	100	ssss	ssss

indexes: types

Depending on the stored keys

dense

key	d1	d2	d3
F	10	null	null
A	1	safaf	ssss
C	1	dddd	ssss
T	100	ssss	ssss
S	10	null	sadfa
Q	1	dsfa	null

position	key
	A
	C
	F
	Q
	S
	T

Non-dense

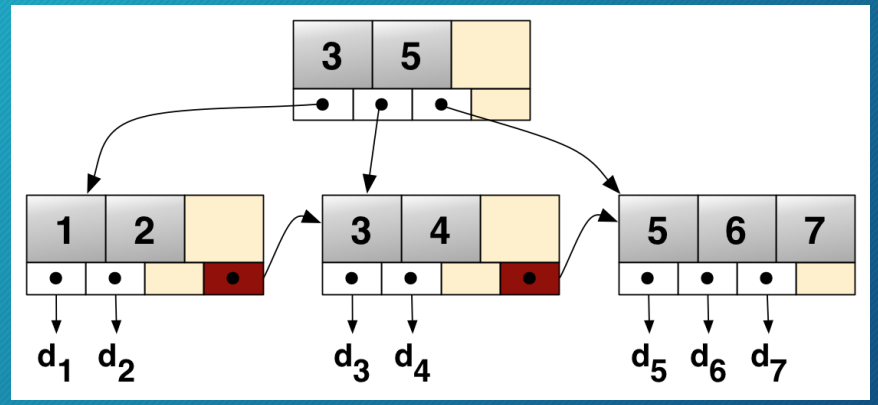
key	position
A-D	
E-M	
N-Z	

key	d1	d2	d3
A	1	safaf	ssss
C	1	dddd	ssss
F	10	null	null
Q	1	dsfa	null
S	10	null	sadfa
T	100	ssss	ssss

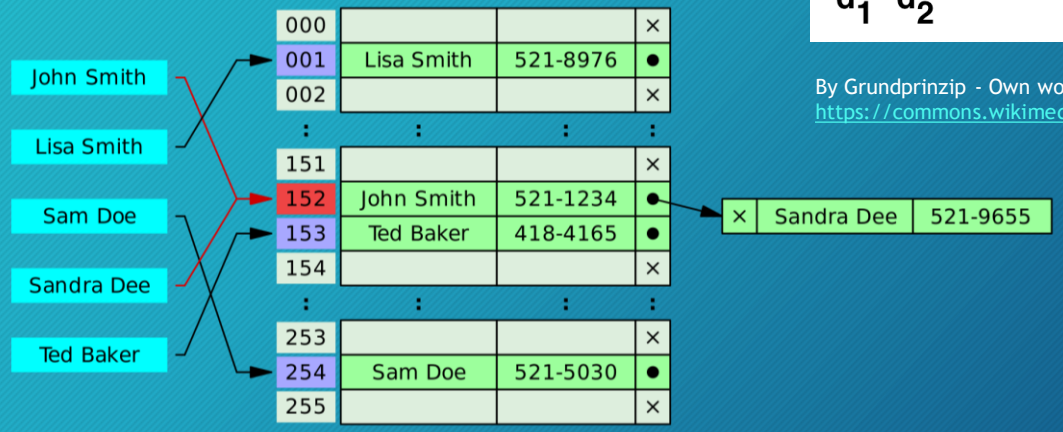
indexes: types

Depending on structure (most common)

- B-Trees
- Hash Tables
- Bitmap



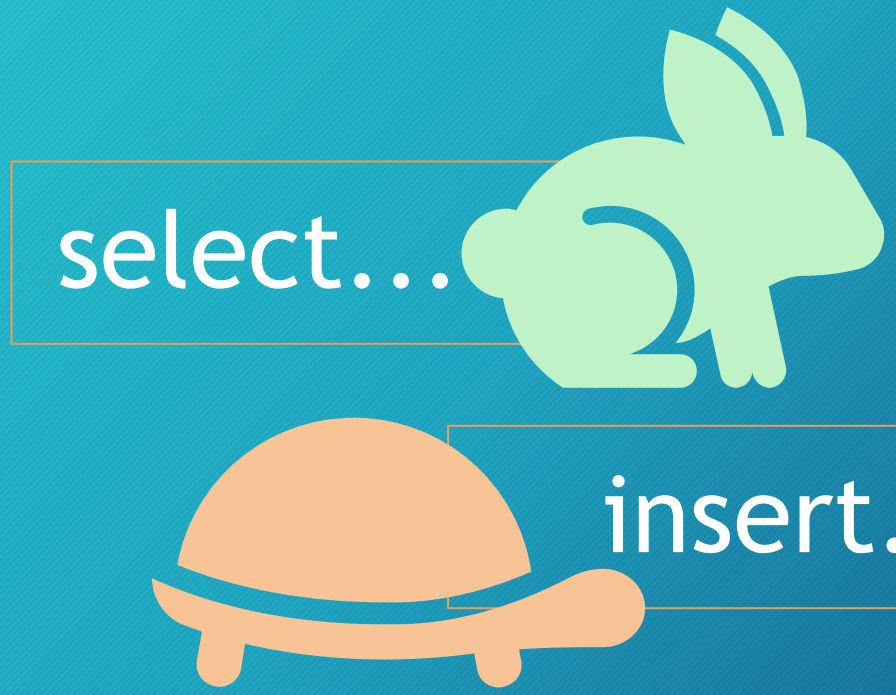
By Grundprinzip - Own work, CC BY 3.0,
https://commons.wikimedia.org/w/index.php?curid=10758840



By Jorge Stolfi - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=6472274

indexes: consequences

Overload due to maintenance of additional structure



clave	posición
A-D	
E-M	
N-Z	

clave	d1	d2	d3
A	1	safaf	ssss
C	1	dddd	ssss
F	10	null	null
Q	1	dsfa	null
S	10	null	sadfa
T	100	ssss	ssss

Indexes: where

- some are automatic (PK)
- Frequent queries

```
select e.nombre
    from empleado e, departamento d
    where d.nombre = 'CONTABILIDAD' and
            e.num = d.num and
            e.edad = 30;
```

```
create index dnom on departamento (nombre);
create index eedad on empleado (edad);
```

Denormalization

NORMALIZATION

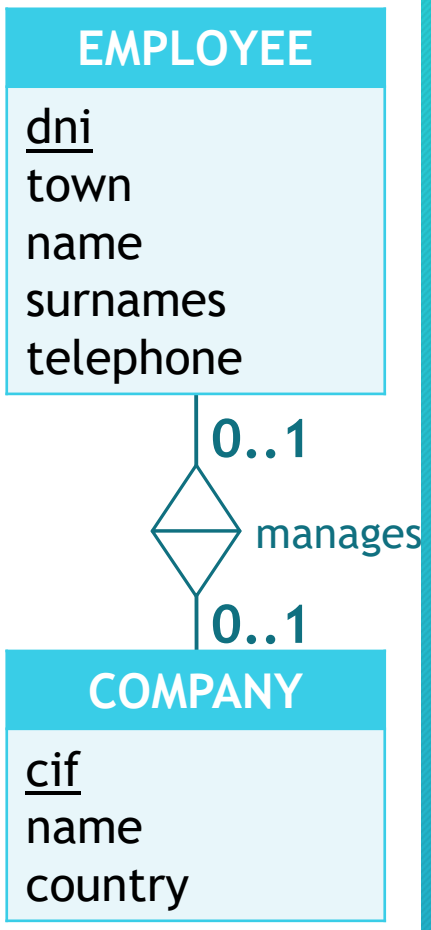
- logical schema structurally consistent and without redundancies

Sometimes performance could improve

"Relaxing" table design

- - benefits from normalized schema
- + performance

denormalization: example




```
CREATE TABLE employee (dni char(12) primary key,
town varchar(75), name varchar(30), surnames varchar(50),
telephone varchar(12));

CREATE TABLE company (cif char(15) primary key,
name varchar(40), country int);


CREATE TABLE manages (
dni char(12) primary key,
cif char(15) NOT NULL UNIQUE,
FOREIGN KEY (dni) REFERENCES employee(dni),
FOREIGN KEY (cif) REFERENCES company(cif));
```

denormalization: example


```
CREATE TABLE employee (dni char(12) primary
key, town varchar(75), name varchar(30), surnames
varchar(50), telephone varchar(12));

CREATE TABLE company (cif char(15) primary
key, name varchar(40), country int);


CREATE TABLE manages (
dni char(12) primary key,
cif char(15) NOT NULL UNIQUE,
FOREIGN KEY (dni) REFERENCES employee(dni),
FOREIGN KEY (cif) REFERENCES company(cif));;
```

dni
21
55
33
42
11
25
30

dni	cif
55	C12
30	C09
21	C11

cif
C11
C12
C09
C20
C19
C18
C23

```
select d.dni,d.nombre,
e.cif, e.nombre
from employee d,
company e, manages g
where d.dni=g.dni
and g.cif=e.cif
```

2 index searching

2 data mixing

denormalization: example

CREATE TABLE employee (dni char(12) primary key, town varchar(75), name varchar(30), surnames varchar(50), telephone varchar(12),
cif char(15) UNIQUE REFERENCES company(cif));

CREATE TABLE company (cif char(15) primary key, name varchar(40), country int);

dni	cif
21	C11
55	C12
33	null
42	null
11	null
25	null
30	C09

cif
C11
C12
C09
C20
C19
C18
C23

```
select d.dni,d.name,
e.cif, e.name
from employee d,
company e
where d.cif=e.cif
```

denormalization: example


```
CREATE TABLE empleado (dni
char(12) primary key, población varchar(75),
nombre varchar(30), apellidos varchar(50), teléfono
varchar(12),
cif char(15) UNIQUE
REFERENCES empresa(cif));

CREATE TABLE empresa (cif char(15) primary key,
nombre varchar(40), país int);
```

dni	cif
21	C11
55	C12
33	null
42	null
11	null
25	null
30	C09

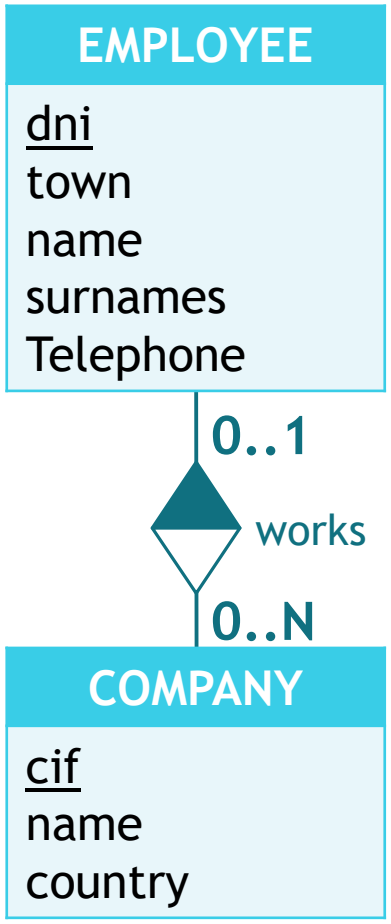
cif
C11
C12
C09
C20
C19
C18
C23

```
select d.dni,d.nombre,
e.cif, e.nombre
from empleado d,
empresa e
where d.cif=e.cif
```

Not always!!!

- table sizes?
- how many relationships?
- Frequency of the query?
- System overload?

denormalization: example

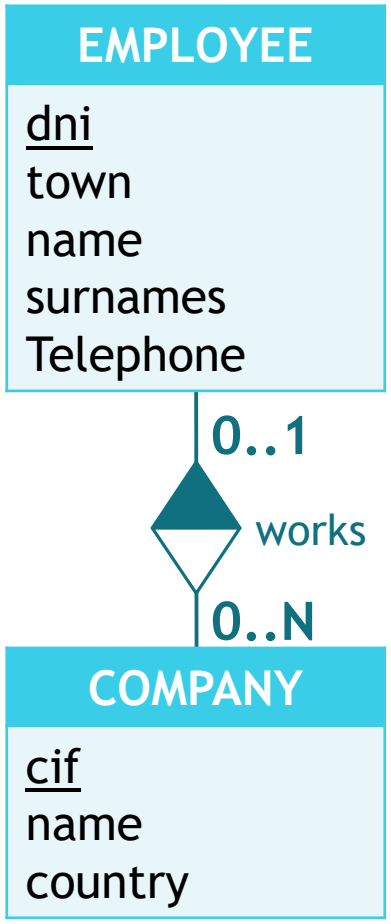



```
CREATE TABLE company (cif char(15) primary key, name
varchar(40), country int);

CREATE TABLE employee (dni char(12) primary key,
town varchar(75), name varchar(30), surnames varchar(50),
telephone varchar(12), cif char(15),
FOREING KEY (cif) REFERENCES company (cif));
```

```
select d.dni,d.name,e.cif,e.name
from employee d, company e
where d.cif=e.cif
```

denormalization: example




```
CREATE TABLE company (cif char(15) primary key, name varchar(40),
country int);

CREATE TABLE employee (dni char(12) primary
key, town varchar(75), name varchar(30), surnames
varchar(50), telephone varchar(12), cif char(15), namempresa
varchar(40),
FOREING KEY (cif) REFERENCES empresa(cif));
```

```
select dni,name,cif,namempresa
from employee
```

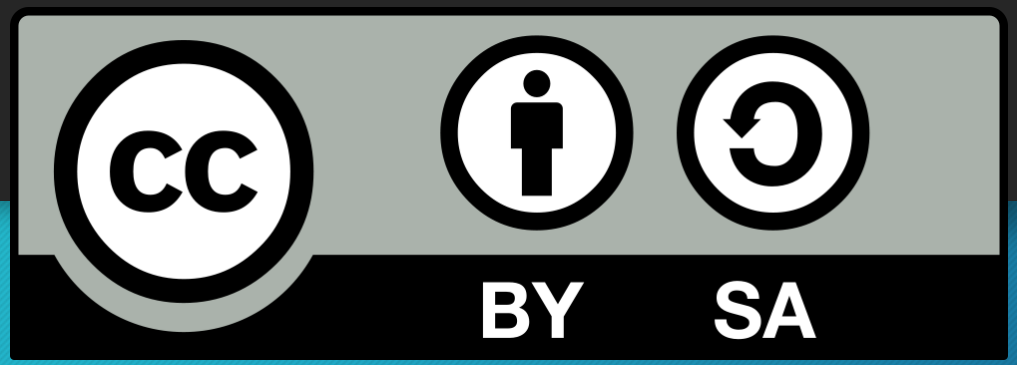
conclusion

- physical design → implementation → system in production
- Schema in a concrete DBMS
- monitoring physical parameters
  - storage
  - performance
- redesign, denormalization...

conclusion: references

- fbddocs.dlsi.ua.es/lecturas
- pixabay.com

licences



lsi Departamento de Lenguajes y Sistemas Informáticos
Universitat d'Alacant
Universidad de Alicante