PROGETTAZIONE CONCETTUALE

- I vincoli si distinguono in due macro-tipologie:
  1) Vincoli **intrarelazionali**:
     - vincoli di dominio
     - vincoli di ennupla
     - vincoli di chiave

  2) Vincoli **interrelazionali**
     - vincoli di integrita' referenziale

- **superchiave:** *Un insieme K di attributi e` superchiave di una relazione r se non contiene due ennuple distinte t1 e t2 tali che t1[K] = t2[K].*

- **chiave:** *Un insieme K di attributi e` chiave di una relazione r se e una ` superchiave minimale per r, ovvero non esiste alcun insieme S tale che S sia superchiave di r e S ⊂ K.*

- **Chiave primaria:** *chiave su cui non sono ammessi valori nulli.*

- **Dipendenza funzionale:** *Una dipendenza funzionale, denotata X → Y , tra due insiemi di attributi X e Y che siano sottoinsiemi di una relazione R specifica un vincolo sulle ennuple che possono formare uno stato di relazione r di R. Il vincolo stabilisce che, per ogni coppia di ennuple t1 e t2 in r per cui t1[X] = t2[X], si ha t1[Y] = t2[Y], ovvero t1[X] = t2[X] → t1[Y] = t2[Y].*

- **PRIMA FORMA NORMALE:**
  - Uno schema di relazione R(X) è detto in **prima forma normale** (1NF o flat) se ogni attributo appartenente a X e un attributo semplice. Altrimenti è detto in forma **nested**
  - In presenza di attributi strutturati o multivalore occorre sostituire l'attributo strutturato con attributi atomici e l'attributo multivalore con una relazione separata che contenga la chiave primaria della relazione originaria.

- **SECONDA FORMA NORMALE:**
  - Uno schema di relazione R e in **2NF** se ogni attributo non primo ` A di R dipende funzionalmente in modo completo dalla chiave primaria di R

- **TERZA FORMA NORMALE:**
  - Uno schema di relazione R e in **3NF** se soddisfa la 2NF e nessun attributo non primo di ` R dipende in modo transitivo dalla chiave primaria

- **BOYCE-CODD:**
  - Uno schema di relazione R e in BCNF se, ogni volta che sussiste in ` R una dipendenza funzionale non banale X → A, X e una superchiave di R

- **Entità** = *Rappresenta una classe di oggetti aventi proprietà comuni ed esistenza 'autonoma' ai fini ` dell'applicazione di interesse.*
- **Relazione** = Rappresenta un legame logico tra due o più entità, significativo per l'applicazione di interesse. Possono essere:
    - Binarie
    - Ricorsive
    - Ternarie
    - Multiple
- **Attributo** = *Descrive una proprietà elementare delle entità e delle associazioni di interesse ai fini dell'applicazione.*

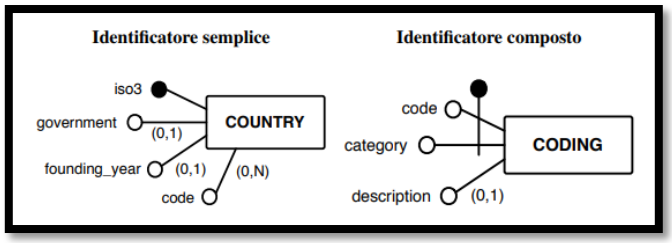
- **Cardinalità** = *I vincoli di cardinalità esprimono le regole che devono regolare la partecipazione delle istanze ` di un' entità a una relazione con altre entità*
    - **Minima =** numero minimo di istanze di associazione R a cui un'istanza di E può partecipare. È detto anche vincolo di partecipazione perché specifica se l'esistenza di un'istanza di entità dipende dal suo essere correlata a un'altra istanza di entità attraverso un'associazione.

    - **Massima =** *Numero massimo di istanze dell'associazione R a cui una istanza dell'entità E può partecipare.*

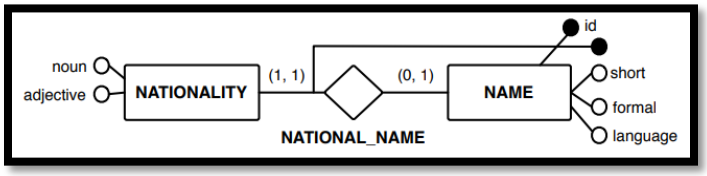
    - In base ai valori di cardinalità massima MC si hanno le seguenti tipologie di relazioni binarie:
        - **1:1 (Uno-a-uno)**: MC = 1 per entrambe le entità E1 e E2
        - **1:N (Uno-a-molti)**: MC = 1 per un'entità e MC = N per l'altra
        - **N:M (Molti-a-molti)**: MC = N per entrambe le entità E1 e E2

    - Anche gli attributi possono avere cardinalità: Descrivono il numero minimo e massimo di valori dell'attributo associati ad ogni istanza di entità o associazione.

- **Identificatore:** Un identificatore di un' entità e una collezione di attributi e/o entità connesse ad E che permettono di identificare univocamente le istanze di E.
    - Un **identificatore interno** e costituito esclusivamente da attributi dell' entità identificata. Pu ` o` essere semplice (costituito da un solo attributo) o composto (costituito da più attributi).



    - **Identificatore esterno** si utilizzano entita con cui E ha un vincolo di dipendenza (i.e., si considerano tipi di associazioni binarie a cui E partecipa con cardinalita (1,1)).

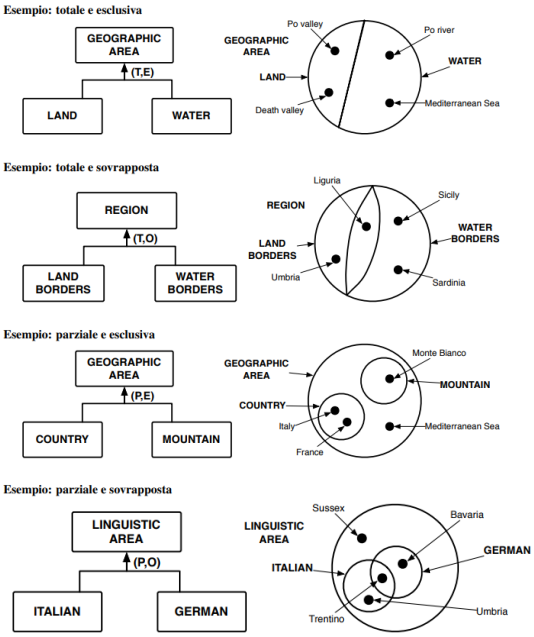


- **Gerarchia di generalizzazione:** Una gerarchia di generalizzazione e un particolare tipo di relazione che intercorre fra un' entità E e altre entità E1, E2, . . . , En. Vi sono alcuni vincoli sulle gerarchie:

  - **<u>Totalità/parzialità</u>:** una gerarchia e **totale** se ogni entità della superclasse e istanza di almeno una sottoclasse della specializzazione; altrimenti è **parziale**

  - **<u>Disgiunzione/Sovrapposizione</u>:** una gerarchia e **esclusiva** (o disgiunta) se un'istanza di entità può essere istanza al più di una delle sottoclassi della specializzazione; altrimenti è **sovrapposta**.

  - **Da ciò derivano le seguenti tipologie di gerarchia:**
    1) **Totale Esclusiva**
    2) **Totale Sovrapposta**
    3) **Parziale Esclusiva**
    4) **Parziale Sovrapposta**



PROGETTAZIONE LOGICA

- Vi sono due principali fasi:
  - **Ristrutturazione dello schema ER**: definizione di un nuovo schema ER ristrutturato eliminando opportunamente le gerarchie di generalizzazione e gli attributi composti e multivalore
  - **Traduzione dello schema ER:** definizione dello schema relazionale risultante dallo schema ER ristrutturato.

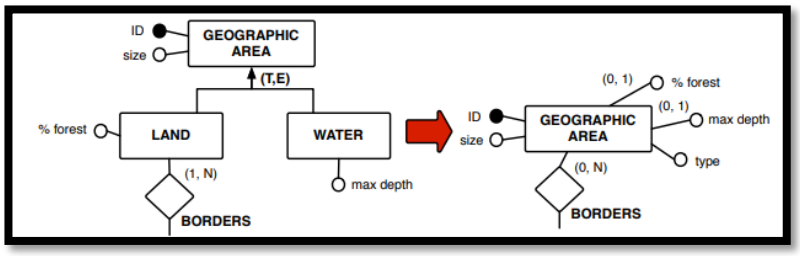
1) Ristrutturazione

- **Attributi derivati:** La presenza di attributi il cui valore e derivato (es., valore aggregato) può essere evitata con l'obiettivo di calcolare il valore derivato solo quando necessario mediante algoritmi a partire da altri valori memorizzati nella BD. In somma alcuni dati possono essere calcolati, non serve che vengano mantenuti nella relazione, ovviamente se il calcolo richiede molte operazioni laboriose conviene tenerlo, ma se le operazioni sono semplici e fatte solo in alcuni casi allora conviene non inserirlo come attributo ma lasciare che esso venga calcolato

- **Gerarchie:** alcuni modelli logici non supportano le gerarchie, bisogna spesso eliminarle:

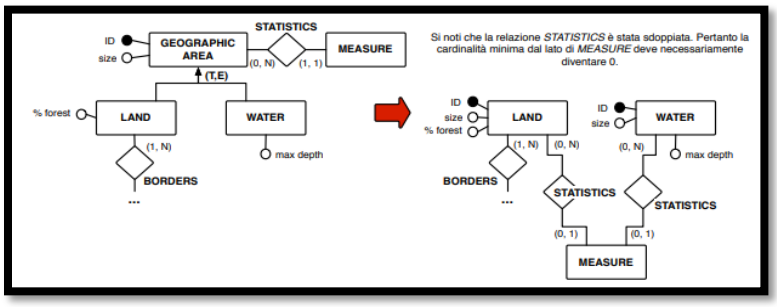
a. Si mantiene solo l' entità padre

- si applica a tutte le gerarchie



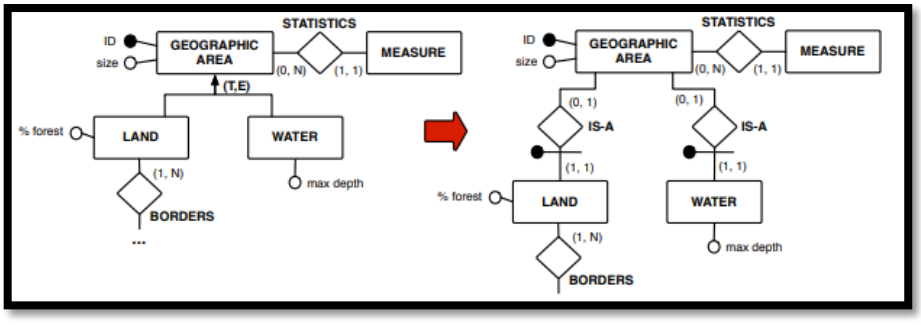
b. si mantengono solo le entità figlie

- si applica a gerarchie totali ed esclusive

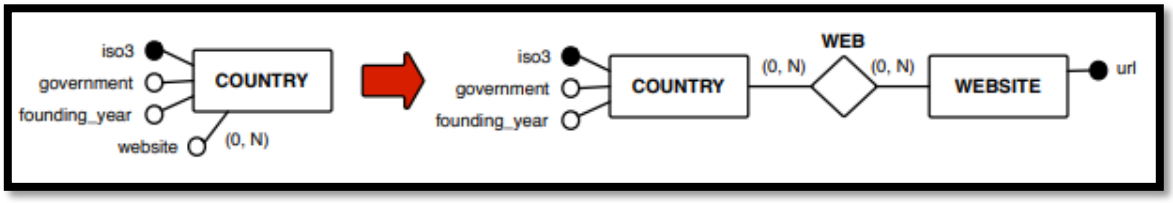


c. si mantengono tutte le entità

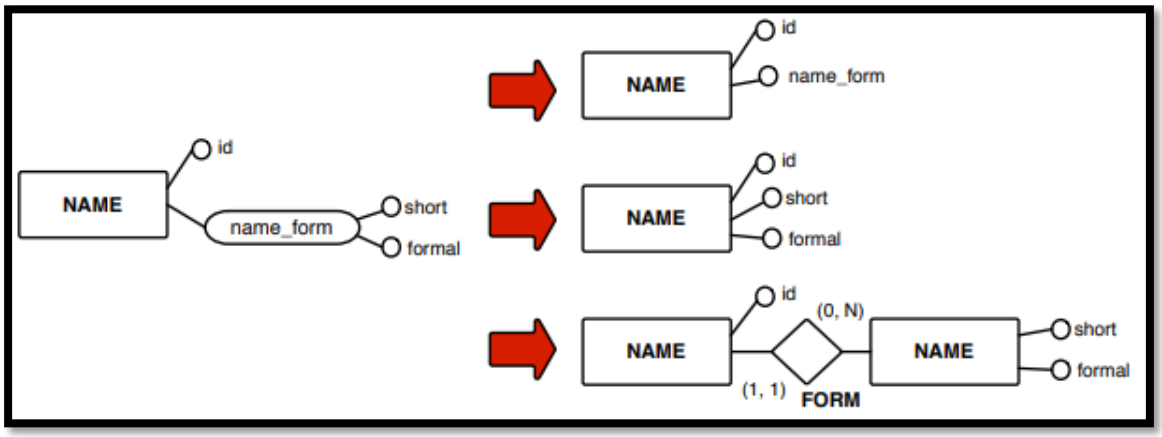
- si applica a tutti i tipi di gerarchia



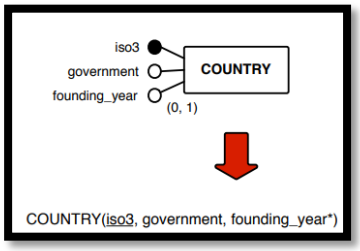
- **Attributi multivalore:** Nel caso di attributi multivalore di un' entità E, si procede a reificare l'attributo in un' entità E_a e si introduce una nuova relazione R fra E e E_a, la cui cardinalità e uguale alla cardinalità dell'attributo multivalore.



- **Attributi composti:** Nel caso di attributi composti e non multivalore di un'entità E, si procede e possibile procedere semplicemente memorizzando le componenti dell'attributo come un unico attributo o come singoli attributi separati.



2) Traduzione dello schema ER



- **Entità:** Data un' entità E = {K, W} con
  K = {a1, a2, . . . , at} come identificatore
  W = {a(t+1), a(t+2), . . . , a(t + n)} come insieme di attributi descrittivi
  - si traduce in E(a1, a2, . . . , at , a(t+1), a(t+2), . . . , a(t + n))

- **Associazione:**
  - ***Molti a Molti***
    - Data un'associazione R = {C} fra le entità E1 = {K1, W1} e E2 = {K2, W2} in cui:
      - max-card(E1, R) = N, max-card(E2, R) = N
      - C insieme di (eventuali) attributi della relazione R
      - K1 e K2 identificatori di E1 e E2, rispettivamente
      - W1 e W2 attributi descrittivi di E1 e E2, rispettivamente
    - si traduce in R(K1, K2, C)

  - ***Uno a Uno***
    - Data un'associazione R = {C} fra le entità E1 = {K1, W1} e E2 = {K2, W2} in cui:
      - max-card(E1, R) = 1, max-card(E2, R) = 1
      - C insieme di (eventuali) attributi della relazione R
      - K1 e K2 identificatori di E1 e E2, rispettivamente
      - W1 e W2 attributi descrittivi di E1 e E2, rispettivamente
    - si traduce in E1(K1, W1, K2, C) E2(K2, W2) oppure E1(K1, W1) E2(K2, W2, K1, C)

  - ***Uno a Molti***
    - Data un'associazione R = {C} fra le entità E1 = {K1, W1} e E2 = {K2, W2} in cui:
      - max-card(E1, R) = 1, max-card(E2, R) = N (o viceversa)
      - C insieme di (eventuali) attributi della relazione R
      - K1 e K2 identificatori di E1 e E2, rispettivamente
      - W1 e W2 attributi descrittivi di E1 e E2, rispettivamente
    - si traduce in E1(K1, W1, K2, C) E2(K2, W2)

ALGEBRA RELAZIONALE

Operatori insiemistici

1) **UNIONE**
   - $r \cup s = \{t \mid t \in r \vee t \in s\}$
   - costituita dalle ennuple t appartenenti a r o a s (o a entrambe)
   - Due relazioni si dicono compatibili all'unione/differenza se sono dello stesso grado e sono definite sugli stessi attributi.

2) **DIFFERENZA**
   - $r - s = \{t \mid t \in r \wedge \neg(t \in s)\}$
   - costituita dalle ennuple t appartenenti a r e non a s

3) **INTERSEZIONE**
   - $r \cap s = \{t \mid t \in r \wedge t \in s\} = (r \cup s) - (r - s) - (s - r)$
   - costituita dalle ennuple t sia a r sia a s

Operatori Unari

1) **SELEZIONE σ**
   - Restituisce una relazione contenente l'insieme delle sole ennuple della relazione operando che soddisfano particolari condizioni, espresse mediante una formula proposizionale
   - in poche parole decide cosa selezionare da quella tabella, si possono scegliere altre condizioni mediante operatori logici NOT,AND,OR
   - $\sigma_{attributo\ -\ condizione}(Tabella)$

2) **PROIEZIONE π**
   - Restituisce come risultato una relazione contenente l'insieme delle ennuple della relazione operando limitate su particolari attributi
   - Come se fosse una print di attributi di una tabella, facendo la proiezione si proiettano solo le colonne scelte dalla tabella
   - $\pi_{attributo}(Tabella)$

Operatori Binari

1) **Prodotto X**
   - Date due relazioni r = R(X) e s = S(Y) con $X \cap Y = \emptyset$, il prodotto cartesiano $r \times s$ e una relazione definita su XY contenente l'insieme delle ennuple che sono combinazione delle ennuple di r e s.
   - In generale e necessario operare una selezione sul risultato del prodotto cartesiano per sele- ` zionare un sottoinsieme significativo di tuple

2) **Theta Join ⋈ Θ**
   - Date r = R(X) e s = S(Y), con X e Y disgiunti, il Theta-Join è una relazione definita su XY composta dalle ennuple risultato della concatenazione di ennuple di r con ennuple di s che soddisfano le condizioni di confronto AΘB (con $A \in X$ e $B \in Y$).
   - $R \bowtie_{A=B} = \sigma_{A=B}(R \times S)$

3) Outer Join

- L'operazione di outer join appende alle ennuple risultato del join anche quelle ennuple di una o entrambe le relazioni che non soddisfano i criteri di join, estendendole con opportuni 'NULL'.
- ci sono
  - ***i.*** LEFT OUTER JOIN
  - ***ii.*** RIGHT OUTER JOIN
  - ***iii.*** FULL OUTER JOIN

WEB

url	country	official
governo.it	ITA	False
igmi.org	ITA	False
stat.it	ITA	False
poste.it	ITA	False

WEBSITE

url	title
governo.it	Governo Italiano
igmi.org	NULL
stat.it	Statistics
poste.it	Poste Italiane
google.com	Google

WEB RIGHT OUTER JOIN WEBSITE

url	country	official	title
governo.it	ITA	False	Governo Italiano
igmi.org	ITA	False	NULL
stat.it	ITA	False	Statistics
poste.it	ITA	False	Poste Italiane
google.com	NULL	NULL	Google

4) Divisione

- La divisione si applica solo a relazioni r(Z), s(X) in cui X ⊆ Z. Dato l'insieme di attributi Y = Z − X, la divisione r ÷ s e una relazione T(Y) contenente tutte le tuple t tali che: - vi siano tuple t_r in r con $t_r[Y] = t$, e $t_r[X] = t_s$ per ogni tupla t_s di s.

r

A	B
x	a
y	a
x	b
z	b

÷

s

B
a
b

=

$r \div s$

A
x

Esempio

Trovare le misure che sono disponibili per tutte le nazioni.

MEASURE

id	value	label	country
952	66028467.0	Population	FRA
957	2.7390000001e+13	GDP	FRA
1345	59831093.0	Population	ITA
1350	2.0680000001e+13	GDP	ITA
2848	64097085.0	Population	GBR

COUNTRY

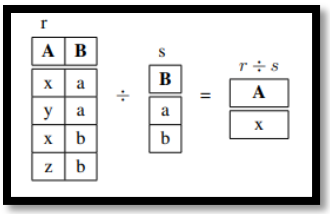
iso3	government
GBR	constitutional monarchy
FRA	republic
ITA	republic

$$(\rho_{country \to iso3}(\Pi_{label,country}(MEASURE)) \div \Pi_{iso3}(COUNTRY)$$

label
Population

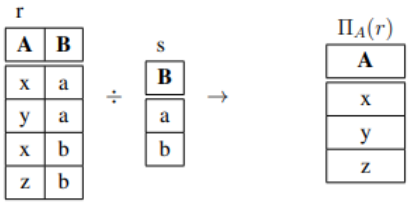
Come fare la divisione passo per passo:

Consideriamo le seguenti tabelle:

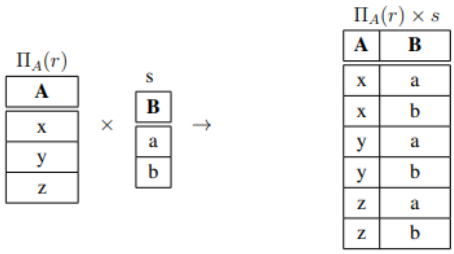


Per effettuare la divisione bisogna:

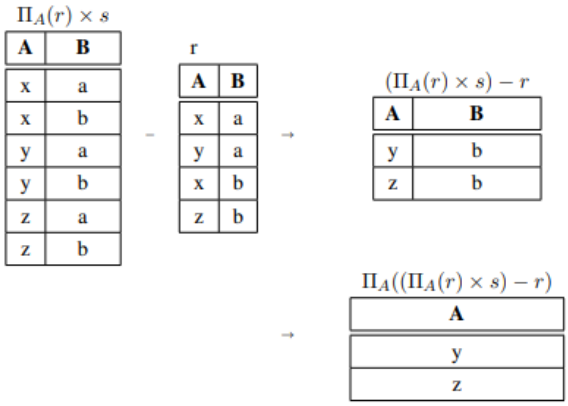
1) Proiettare gli attributi **non** in comune alle due relazioni



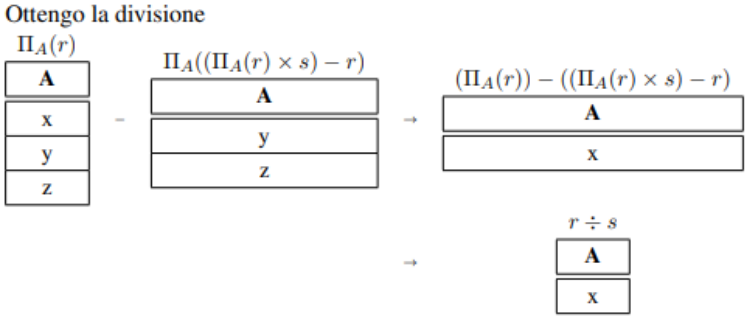
2) Definire tutte le possibili combinazioni fra gli elementi in A e in s



3) Togliere alle possibili combinazioni le combinazioni effettive in r



4) Si ottiene la divisione



APPUNTI GENERALI SU SQL

- L'operatore aggregato necessita di lavorare su dati aggregati.
    - Necessita di applicare operatori aggregati a sottogruppi di tuple di una relazione in base al ` valore di uno o piu attributi. → GROUP BY
    - Prima si effettua il raggruppamento e poi si applica l'operatore aggregato a ciascun sottogruppo individuato.
    - **HAVING specifica una condizione su un gruppo di tuple associata al valore degli attributi di raggruppamento.**
    - COUNT(*) conta le righe, mentre COUNT(attributo) conta i valori di 'attributo'

- Le operazioni insiemistiche UNION, INTERSECT, EXCEPT sono state incorporate direttamente in SQL.
    - **UNION**: arricchisce la potenza espressiva di SQL e permette di scrivere interrogazioni altrimenti non formulabili. UNION ALL preserva i duplicati
    - **Es:** Determinare l'insieme dei nomi brevi e formali delle nazioni.

```
SELECT short FROM name
UNION
SELECT formal FROM name;
```

- È possibile definire subquery che sono eseguite ripetutamente per ogni tupla candidata considerata nella valutazione della query esterna → **QUERY CORRELATE**
    - Per poter far riferimento alle colonne delle tuple candidate nella query esterna si fa uso degli alias di relazione.
    - Es: *Determinare le nazioni (e il corrispondente tipo di governo) il cui prodotto lordo superiore alla media del prodotto delle nazioni che hanno la stessa forma di governo*

```
SELECT M.country, C.government
FROM country AS C JOIN measure AS M ON C.iso3 = M.country
WHERE M.label='GDP' AND M.value >= (
  SELECT AVG(M1.value)
  FROM country AS C1 JOIN measure AS M1 ON C1.iso3 = M1.country
  WHERE M1.label='GDP' AND C1.government = C.government
)
```

- Una **vista** in SQL e una tabella derivata a partire da altre tabelle (tabelle di base o altre viste).
    - Una vista è una tabella virtuale: non ci sono tuple istanza della vista memorizzate nella BD
    - Anche se virtuale, una vista puo essere utilizzata nella formulazione di interrogazioni come ` se fosse memorizzata

PROGETTAZIONE FISICA

- I dati sono memorizzati sotto forma di record. Ogni record è una collezione di valori di dati collegati in cui ogni valore e un insieme di uno o più byte e prende il nome di ` campo del record. I record possono essere:
    - A lunghezza **FISSA**: Se i record hanno tutti la stessa dimensione in byte
    - A lunghezza **VARIABILE**: Se i record non hanno tutti la stessa dimensione in byte

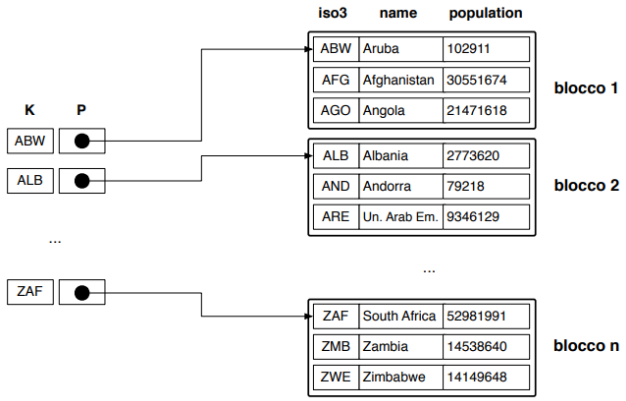
- I record vanno ripartiti in blocchi
    - Un blocco dimensionato B Byte
    - Un record composto da R (<=B) Byte
    - Allora si possono inserire bfr records dove $bfr = \left\lfloor \frac{B}{R} \right\rfloor$
    - Ciò comporta l'esistenza di *B – (bfr ∗ R) byte*

- I record vengono memorizzati in File, essi possono essere
    - **ORDINATI**: Inserimento e cancellazione lenti ma ricerca veloce
    - **NON ORDINATI**: Inserimento e cancellazione veloci ma ricerca lenta

- Un **INDICE** è una **struttura di accesso secondaria** utilizzata per rendere più veloce l'accesso ai dati memorizzati in una delle strutture fisiche descritte in precedenza
    - velocizza l'accesso, ma ha un costo di mantenimento in inserimento, cancellazione e modifica
    - Gli indici possono essere:
        - **PRIMARIO**: specificato sul campo chiave di ordinamento di un file ordinato di record. Il campo di ordinamento e tale che tutti i record hanno ` valori univoci per quel campo
        - **CLUSTER**: se si vuole indicizzare un campo con valori non univoci (i.e., non chiave) si usa un indice di cluster su file chiamati file clustered
    - Si noti che un file può avere al massimo un campo di ordinamento fisico, perciò può **avere al massimo un indice primario o un indice di cluster, ma non entrambi**. Su qualsiasi campo non di ordinamento di un file si può invece definire un indice secondario.
    - Gli indici possono essere
        - **DENSI**: contenere una voce per ogni valore della chiave di ricerca
        - **SPARSI** contenere voci solo per alcuni valori della chiave di ricerca

INDICE PRIMARIO

- **n voci dell'indice primario = n blocchi nel file**
- primo blocco = "blocco ancora"
- sono indici **SPARSI** poiché un blocco contiene più record



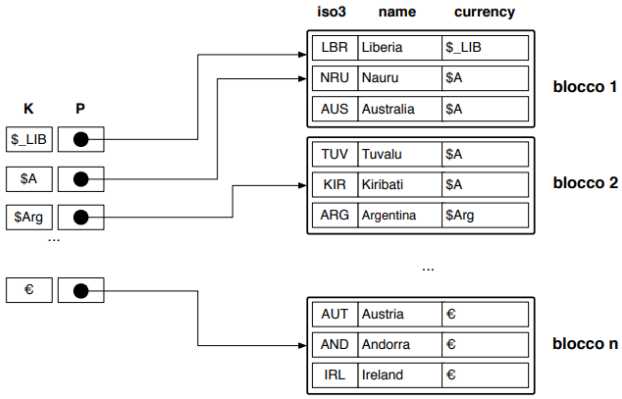
- **Esempio:**

Supponiamo di avere un file ordinato:

- Record a dimensione fissa $R = 100$ byte e indivisibili.
- File con $r = 30.000$ record e blocchi $B = 1024$ byte.
- Fattore di blocco: $bfr = \lfloor B/R \rfloor = \lfloor 1024/100 \rfloor = 10$ record per blocco.
- Il numero di blocchi necessari per il file è $b = \lceil r/bfr \rceil = \lceil 30.000/10 \rceil = 3000$ blocchi.
- Una ricerca binaria sul file di dati richiederebbe approssimativamente $\lceil \log_2 b \rceil = \lceil \log_2 3000 \rceil = 12$ accessi al file.

INDICE DI CLUSTER

- usati per **file ordinati secondo un campo di ordinamento non chiave**
- la **chiave dell'indice contiene una voce per ogni valore distinto del campo di ordinamento**. Il puntatore si riferisce al primo blocco che contiene almeno un record corrispondente alla chiave dell'indice
- è <u>SPARSO</u>



INDICE SECONDARIO

- **per esserci deve esistere anche un indice primario**
- fornisce un'ulteriore struttura di accesso ad un file
- il file dei record può essere ordinato o non ordinato
- il campo indicizzato può essere una chiave oppure no
- **Se il campo di indicizzazione è chiave, l'indice contiene tutti i valori del campo come chiave dell'indice e puntatori al blocco in cui è memorizzato il record o puntatori al record stesso**.
    - **→ L'indice risulta DENSO**
- L'indice contiene molte voci (una per ogni valore del campo di indicizzazione) ma **il risparmio in termini di tempo di ricerca è comunque notevole** perché il campo di indicizzazione non è campo di ordinamento e perciò una ricerca senza indice non consentirebbe la ricerca binaria.
- **La chiave dell'indice è mantenuta ordinata → permette una ricerca binaria**
- **Se il campo di indicizzazione può avere valori duplicati, vi sono tre opzioni di memorizzazione dell'indice**:
    - inserire più voci dell'indice con medesimo valore di K
    - usare un record a lunghezza variabile per l'indice in modo da inserire più puntatori per ogni voce dell'indice
    - mantenere voci a lunghezza fissa ma inserendo un ulteriore livello per i puntator

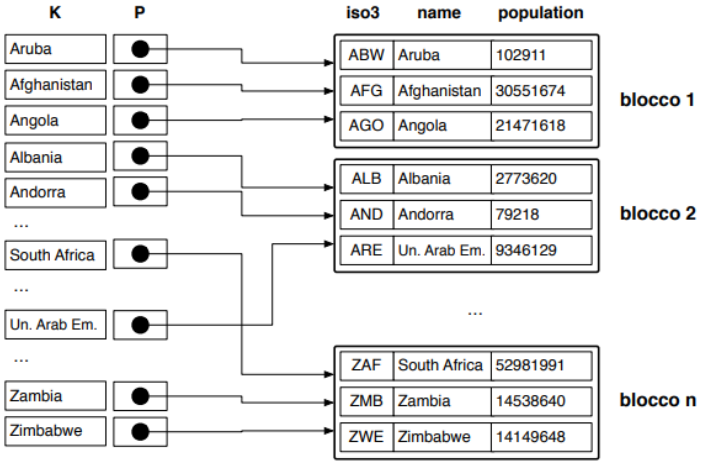


Figure 2 Senza valori duplicati

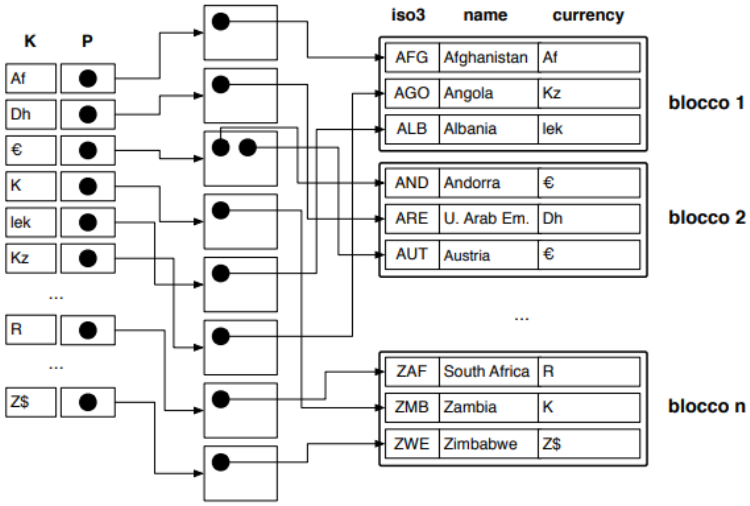


Figure 1 Con valori duplicati

INDICE	Num. Voci	Denso	Oggetto del puntatore
Primario	N. blocchi	No	Blocco
Cluster	N. valori distinti	No	Blocco
Secondario Chiave	N. Record	Si	Record/blocco
Secondario non chiave	N. Record	Si/No	Record