Quindi, per poter confondere un individuo con altri k, posso utilizzare queste due tecniche, ma i dati perdono utilità. Bisogna trovare un compromesso tra utilità e privacy. I dati sanitizzati, cioè dati privati di attributi sensibili, spesso, perdono di utilità.

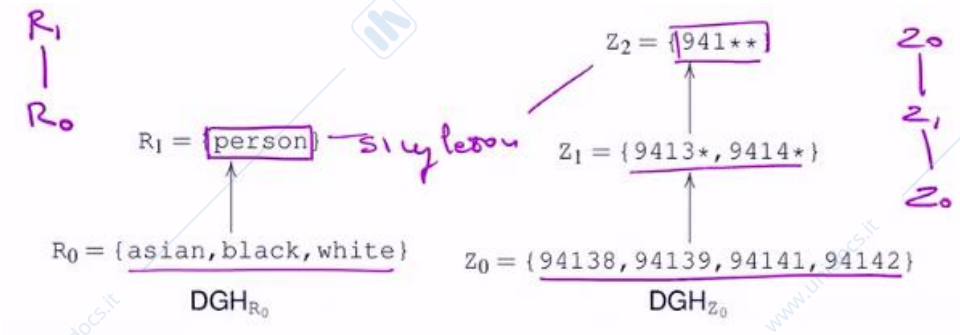
Domain generalization hierarchy

Posso costruire su ciascun dominio una gerarchia di generalizzazione che ti dice come andare dai valori più specifici ai valori più generici. Ad esempio le date a livello completo giorno mese e anno rispetto alla data nella forma mese anno, li metto quindi in gerarchia dove dico che un dominio è una specializzazione di un altro e un dominio è una generalizzazione dell'altro. Ho quindi una gerarchia di generalizzazione sui miei domini

- Given two domains $D_i, D_j \in \text{Dom}$, $D_i \leq_D D_j$ states that the values in domain D_j are generalizations of values in D_i

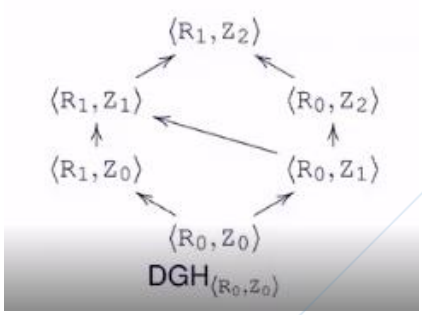
  - $\forall D_i, D_j, D_z \in \text{Dom}$: $D_i \leq_D D_j, D_i \leq_D D_z \Longrightarrow D_j \leq_D D_z \vee D_z \leq_D D_j$

Sta cosa sopra mi dice che i domini sono in relazione d'ordine totale, se ad esempio D_i è dominato da D_j e da D_z allora abbiamo che Dz domina Dj che domina Di oppure Dj che domina Dz che domina Di però tra dj e dz c'è una relazione d'ordine che vuol dire che i domini sono in relazione d'ordine totale. Inoltre la radice della tua catena è un singleton, esempio quando vai a generalizzare fino all'ultimo livello ad esempio il CAP, tutti i CAP saranno uguali. Esempio:



Gerarchia di generalizzazione per l'attributo razza a sx e per l'attributo zip a dx

Se ho un quasi identifier composto da razza e zip insieme ho 6 possibili generalizzazioni:

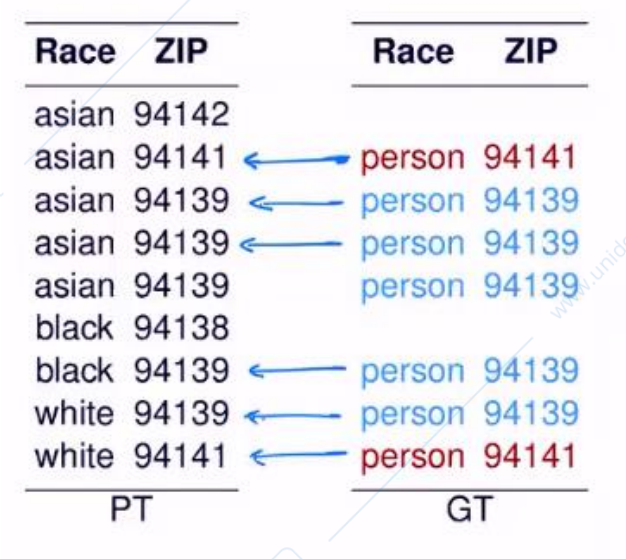


Posso generalizzare o meno sia la razza che lo zip.

Generalized table with suppression

Se ho due tabelle Ti e Tj , Tj è una versione generalizzata di Ti al più sopprimendo qualche tupla se:

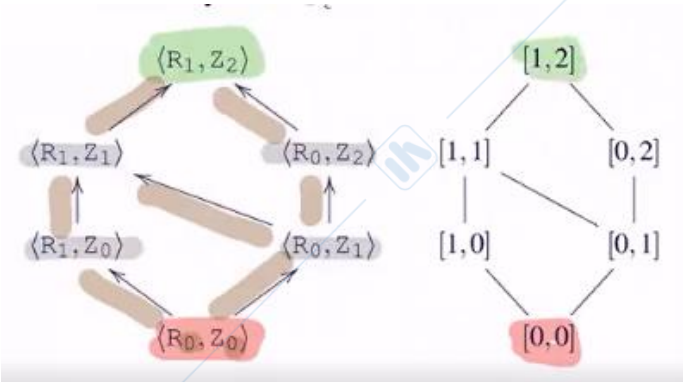
1) |Tj|<= |Ti|, la cardinalità di Tj è uguale al più alla cardinalità di Ti
2) Per ogni attributo il valore che c'è nella colonna è una generalizzazione dei valori che c'erano nella tabella originale
3) È possibile definire una funzione che ti mappa ogni tupla della tabella generale ad una tupla della tabella specifica, questa funzione è iniettiva, esempio:



Non tutte le tuple di partenza saranno raggiunte perché alcune sono state soppresse.

Quando necessito di sanitizzare una tabella è meglio sopprimere oppure generalizzare? Se ho poche tuple diverse dalle altre conviene cancellare la tupla per evitare di generalizzare tutti gli altri. Quindi con pochi outlier conviene cancellare piuttosto che generalizzare troppo le tuple restanti. Ad esempio, un novantenne curato in una situazione di emergenza in un ospedale pediatrico, è più efficiente cancellare la tupla corrispondente all'uomo di 90 anni dalla pubblicazione statistica, piuttosto che generalizzare tutte le date di nascita.

Il **Distance vector** ha un elemento per ciascun dominio e mi mette come valore i passi di generalizzazione che io ho fatto rispetto a quel dominio.



K minimal generalization with suppression

Se ho due tabelle Ti e Tj, dove Tj è la generalizzazione di Ti, supposto che tu puoi sopportare un certo livello di soppressione (es fino all'1% delle tuple o fino al 2%) MaxSup, Tj è una soluzione minimale se:

1)Fa la minima soppressione che serve, non ci sono altre tabelle con lo stesso vettore di distanza che hanno lo stesso grado di k anonymity, cancellando di meno.

2) |Ti| - |Tj| <MaxSup, non sopprimi più di quanto ti ho dato il permesso di sopprimere

3) $\forall T_z : T_i \preceq T_z$ and T_z satisfies conditions 1 and 2 $\Longrightarrow \neg(DV_{i,z} < DV_{i,j})$

Ovvero per ogni Tz, dove Tz è una generalizzazione di Ti , e Tz mi soddisfa le prime due condizioni, non è vero che il distance vector di i,z è dominato dal distance vector i,j; Non esiste una generalizzazione più specifica che fa lo stesso lavoro di quella scelta, se no sceglierei direttamente quella. Ciò significa che, a parità di requisiti, scelgo quello che generalizza di meno.

MaxSup=2

[1,0] [0,1]

Race:R_0	ZIP:Z_0
asian	94142
asian	94141
asian	94139
asian	94139
asian	94139
black	94138
black	94139
white	94139
white	94141

PT

Race:R_1	ZIP:Z_0
person	94141
person	94139
person	94139
person	94139
person	94139
person	94139
person	94141

$GT_{[1,0]}$

Race:R_0	ZIP:Z_1
asian	9414*
asian	9414*
asian	9413*
asian	9413*
asian	9413*
black	9413*
black	9413*

$GT_{[0,1]}$

Tra questi sopra non c'è un meglio e un peggio, ci sono dei criteri per stabilire quale è meglio

• Il criterio di preferenza distanza assoluta minima (minimum absolute distance) preferisce la generalizzazione (s) con la distanza assoluta più piccola, ovvero con il numero totale più piccolo di passaggi di generalizzazione (indipendentemente dalle gerarchie su cui sono state eseguite).

• Il criterio di preferenza distanza relativa minima (minimum relative distance) preferisce la generalizzazione (s) con la distanza relativa più piccola, cioè che minimizza il numero totale di passi relativi (un passo viene reso relativo dividendolo per l'altezza della gerarchia di domini a cui si riferisce (drel = n.passi/n.passitot)).

• Il criterio di preferenza massima distribuzione preferisce le generalizzazioni con il maggior numero di tuple distinte.

• Il criterio di preferenza soppressione minima preferisce la generalizzazione (s) che sopprime meno tuple, cioè quella con la più grande cardinalità.

Le tecniche per soddisfare il requisito di k-anonimity sono generalizzazione e soppressione, che possono essere applicate a livelli diverse di granularità:

• La generalizzazione può essere applicata a livello di singola colonna (ovvero, una fase di generalizzazione generalizza tutti i valori nella colonna) o singola cella (ovvero, per una colonna specifica, la tabella può contenere valori a diversi livelli di generalizzazione).

• La soppressione può essere applicata a livello di riga (ad esempio, un'operazione di soppressione rimuove un'intera tupla), colonna (ad esempio, un'operazione di soppressione oscura tutti i valori di una colonna) o singole celle (ad esempio, una tabella anonimizzata k può cancellare solo alcune celle di una data tupla/attributo).

	Suppression			
Generalization	*Tuple*	*Attribute*	*Cell*	*None*
Attribute	**AG_TS**	**AG_AS** ≡ AG_	**AG_CS**	**AG_** ≡ AG_AS
Cell	**CG_TS** not applicable	**CG_AS** not applicable	**CG_CS** ≡ CG_	**CG_** ≡ CG_CS
None	**_TS**	**_AS**	**_CS**	_ not interesting

Figura 10: Tecniche per soddisfare k-anonimity: Generalizzazione e Soppressione

Il problema di trovare tabelle k-anonime minime, con attributo la generalizzazione e la soppressione delle tuple è difficile dal punto di vista computazionale. La maggior parte degli algoritmi esatti proposti in letteratura ha un tempo esponenziale computazionale nel numero degli attributi che compongono il quasi-identificatore: quando il numero |QI| degli attributi nel quasi identificatore è piccolo rispetto al numero n di tuple nella tabella privata PT, questi algoritmi esatti con generalizzazione degli attributi e soppressione della tupla sono pratici. Sono stati proposti molti algoritmi esatti per la produzione di tabelle k-anonime attraverso la generalizzazione degli attributi e la soppressione della tupla.

Algoritmi per AG_TS e AG_

Se tu guardi la gerarchia, ciascun cammino rappresenta una strategia per trovare una soluzione

Definiamo localmente la generalizzazione minima il nodo più basso di ogni percorso che soddisfa l'anonimato k. Le proprietà che vengono sfruttate dall'algoritmo sono:

• Ogni generalizzazione k-minimal è localmente minima rispetto a un percorso (ma il contrario non è vero).

• Salendo nella gerarchia, diminuisce il numero di tuple che devono essere rimosse per garantire l'anonimato k.

Se non esiste una soluzione che garantisca l'anonimato k che sopprima un numero di tuple inferiore a MaxSup relativo all'altezza h, non può esistere una soluzione, con un'altezza inferiore a h che la garantisca. L'algoritmo adotta una ricerca binaria sul reticolo della distanza vettori:

1. Valutare le soluzioni ad altezza ⌊h/2⌋

2. Controllare se esiste almeno una soluzione che soddisfa il requisito di k-anonimity:

• Se esiste, valutare le risoluzioni all'altezza ⌊h/4⌋

• Altrimenti, valutare le soluzioni all'altezza ⌊3h/4⌋

3. Ripetere fino a quando l'algoritmo raggiunge l'altezza più bassa per la quale esiste un vettore di distanza che soddisfi l'anonimato k.

Per ridurre il costo computazionale, adotta una matrice rappresentante i vettori di distanza che evita il calcolo esplicito di ogni tabella generalizzata.

k-optimize

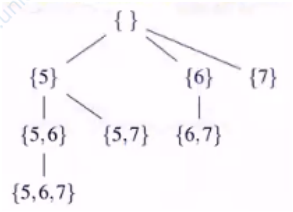
L'algoritmo k-optimize permette di generalizzare attraverso gli indici. K-optimize ordina gli attributi in QI e i valori nei loro domini. Associa un valore di indice (numero intero) a ciascun valore di dominio, seguendo l'ordine definito, mettiamo quindi i valori del dominio in un ordine totale

Race			ZIP			
⟨[asian]	[black]	[white]⟩	⟨[94138]	[94139]	[94141]	[94142]⟩
1	2	3	4	5	6	7

Si creano dei cluster mettendo insieme valori vicini, lavorano sull'ordinamento del dominio:

- Race: {1}, that is: ⟨[asian or black or white]⟩
- ZIP: {4, 6}, that is: ⟨[94138 or 94139],[94141 or 94142]⟩

Race 1 vuol dire a partire da 1, zip 4 6 vuol dire un gruppo da 4 a 6 escluso e un gruppo da 6 in poi. Dopodichè si mettono questi gruppi in un albero, il valore più piccolo nei gruppi si omette.



{5,6,7} vuol dire 4 5 6 7

{5,6} vuol dire 4 5 [6 7]

{5,7} vuol dire 4 [5 6] 7

{6,7} vuol dire [4 5] 6 7

{5} vuol dire 4 [5 6 7]

{6} vuol dire [4 5] [6 7]

{7} vuol dire [4 5 6] 7

{ } vuol dire [4 5 6 7]

Se organizzo così i miei gruppi come trovo la mia soluzione? Sopra k è soddisfatto, io scendo giù nell'albero e continuo fino a che la soluzione c'è, se io arrivo ad un nodo che non soddisfa k anonimity lo elimino facendo il pruning, quindi elimino tutti i figli di quel nodo perché i figli di mio figlio spezzano quello che era insieme a lui in gruppi diversi, quindi se mio figlio non rispetta k anonimity anche i figli dei miei figli non lo rispettano. É un albero tale per cui se si osserva il padre che va al figlio spezziamo qualcosa che prima era insieme, mentre se si osserva dal figlio al padre devo unire qualcosa che prima era diviso. La radice è l'elemento più generico, mano a mano che si discende nell'albero si ottengono soluzioni più specifiche. Se un figlio non è k-anonimo, significa che i suoi figli o i suoi nipoti non sono k-anonimi, a quel punto faccio pruning (potare) di quel ramo. k-Optimize visita l'albero (ad es. utilizzando una ricerca depth-first) per cercare l'anonimato con il costo più basso. Poiché il numero di nodi nella struttura è 2 |I| , la visita della struttura non è pratica. Si adotta quindi, come dicevamo precedentemente, una strategia di potatura per ridurre i costi di calcolo. Il nodo n viene eliminato se nessuno dei suoi discendenti potrebbe essere ottimale. Questa determinazione può essere fatta calcolando un limite inferiore sul costo dei nodi nella sottostruttura radicata in n: se il limite inferiore è maggiore del miglior costo attuale, il nodo n viene eliminato.

Algoritmo incognito

Se tu hai un quasi identifier e quello che ti devo produrre è una soluzione che abbia almeno k occorrenze dei quasi identifier, per avere almeno 4 occorrenze di una quadrupla ciascuna tripla dovrà avere almeno 4 occorrenze, ciascuna coppia deve avere almeno 4 occorrenze e ciascun singolo dovrà avere almeno 4 occorrenze, esempio se io devo avere almeno 2 occorrenze per gli attributi Race e zip, per avere almeno 2 occorrenze della coppia ciascun valore che appare dovrà apparire almeno 2 volte.

e se un singolo attributo non ha k-occorrenze, non può avere k-occorrenze in combinazione con altri attributi. Il funzionamento dell'algoritmo può essere riassunto come segue:

• Iterazione 1: controlla l'anonimato k per ciascun attributo in QI, scartando le generalizzazioni che non soddisfano l'anonimato k.

• Iterazione 2: combina le restanti generalizzazioni in coppia e verifica l'anonimato k per ogni coppia ottenuta.

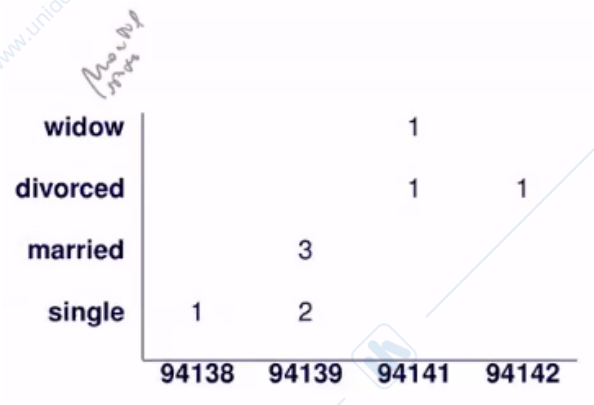
• Iterazione i: considera tutti i-uples di attributi, ottenuti combinando generalizzazioni che soddisfano l'anonimato k all'iterazione i - 1, eliminando le soluzioni non k-anonime.

• Iterazione |QI| restituisce il risultato finale.

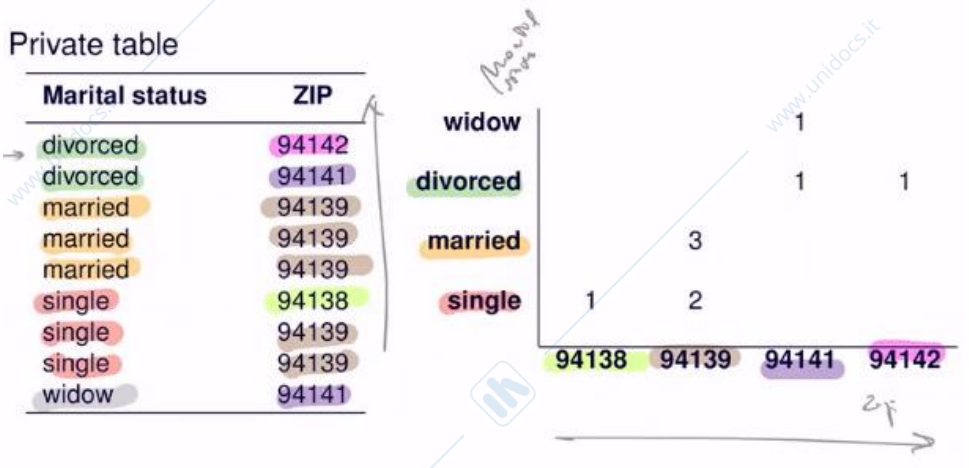
Algoritmi per _CS e CG_

Algoritmo multidimensionale Mondrian

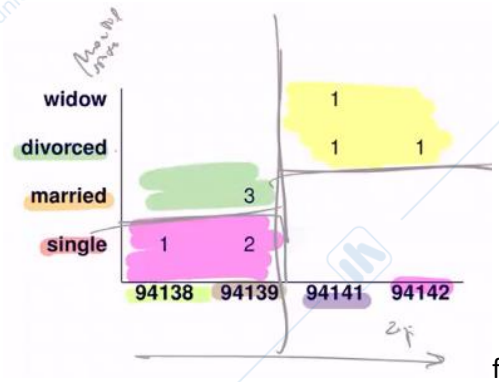
Ogni quasi identifier mi rappresenta una dimensione, es razza e zip 2 dimensioni, razza zip sesso 3 dimensioni etc, ogni tupla rappresenta un punto nello spazio definito da queste dimensioni, tuple con lo stesso valore corrispondono a più punti in quello spazio definito da quelle dimensioni. I valori vengono messi in ordine:



Ciascuna tupla rappresenta un punto nello spazio, quando hai più tuple hai una molteplicità di valori, es



Come trovo la soluzione? Applico gli algoritmi per operare su uno spazio multidimensionale di questo tipo andando a tagliare lo spazio in modo che tu abbia in ciascun taglio che io faccio almeno k punti, ad esempio se vuoi un k anonimity di 3 posso fare una cosa di questo genere:



faccio tagli di spazio

Fare questo taglio vuol dire che vedo il giallo come uno spazio, il verde come uno spazio e il rosa come uno spazio, in ogni spazio ci sono almeno 3 persone, ovvero quello che voglio.

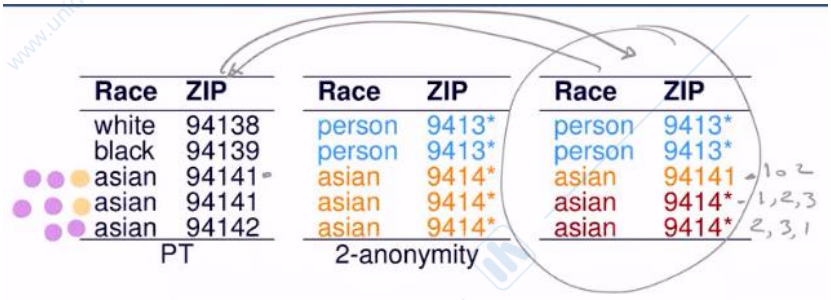
Dal punto di vista di pubblicare i dati vuol dire che una volta che hai tagliato il tuo spazio, quando vai a pubblicare i dati vai a pubblicare i valori generalizzati che corrispondono a quello spazio. Es.

I verdi(married) li posso pubblicare così come sono perché corrispondono a married e 94139, quelli viola sono single ma con 94138 e 94139 e li pubblico o come 94138 o come 94139, quelli gialli corrispondono a widow e divorced e 94141 e 94142 e li pubblicherò come widow e divorced e ancora col dubbio tra 94141 e 94142.

k-anonimity rivisitato

Il requisito di anonimato k consiste nel fatto che ogni rilascio di dati deve essere tale che ogni combinazione di valori di quasi-identificatori possa essere indistintamente abbinata ad almeno k intervistati. Quando la generalizzazione viene eseguita a livello di attributo (AG), ciò equivale a richiedere che ogni quasi-identificatore n-uple abbia almeno k occorrenze. Quando la generalizzazione viene eseguita a livello di cella (CG), l'esistenza di almeno k occorrenze è una condizione sufficiente ma non necessaria; un requisito meno rigoroso sarebbe sufficiente

1) Per ogni sequenza di valori nella tabella privata ci siano almeno k tuple a cui può matchare
2) Per ogni sequenza di valori nella tabella pubblica ci siano almenk k tuple a cui potrebbe matchare.



Asian 94141 potrebbe essere nella terza tabella o l'arancio o uno dei rossi, stessa cosa quello sotto nella prima tabella che è uguale, il terzo potrebbe essere uno dei rossi. Per quanto riguarda la direzione dalla terza tabella alla prima, l'asian arancione potrebbe essere il terzo o il quarto della prima tabella, il primo asian rosso potrebbe essere nella prima tabella o il terzo o quarto o il quinto, mentre l'ultimo asian rosso potrebbe essere uno degli ultimi tre. Rispetto a chi guarda i dati io ho questa incertezza anche se non ho k occorrenze

Race	ZIP
white	94138
black	94139
asian	94141
asian	94141
asian	94142

PT

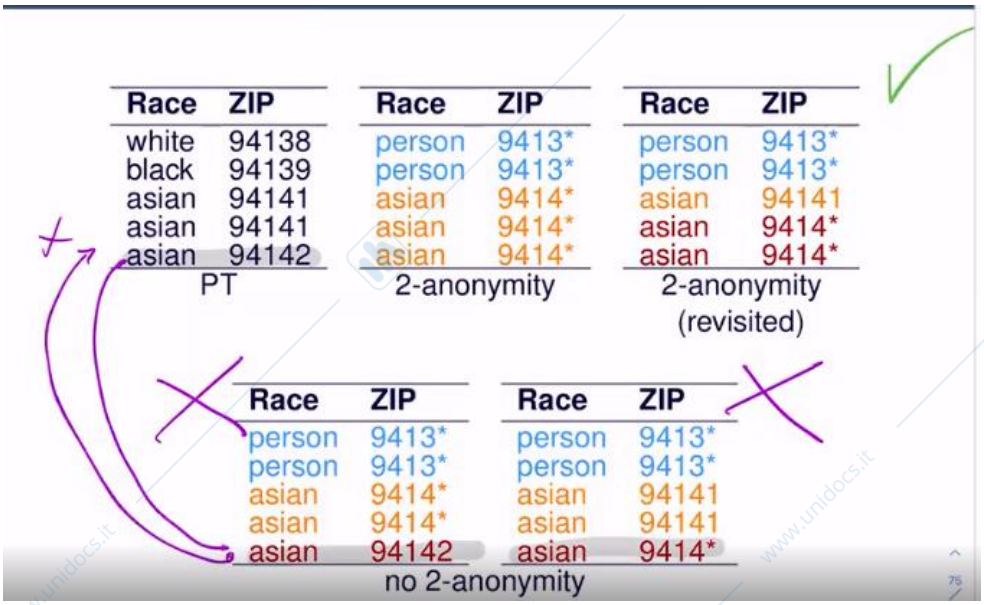
Race	ZIP
person	9413*
person	9413*
asian	9414*
asian	9414*
asian	9414*

2-anonymity

Race	ZIP
person	9413*
person	9413*
asian	94141
asian	9414*
asian	9414*

2-anonymity (revisited)

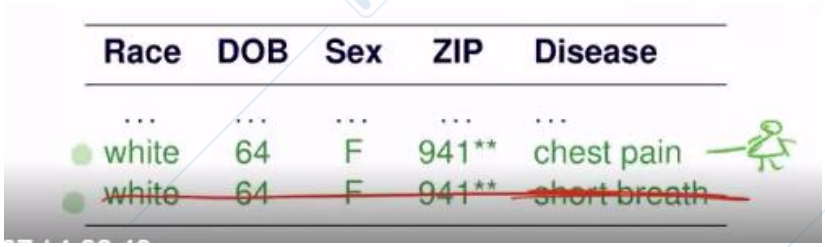
Posso fare queste conclusioni solo se ho la tabella di partenza, il vantaggio è che posso rilasciare tabelle guardando i dati che ho dal lato mio, non ho bisogno di conoscere i dati fuori, prendo quindi l'assunzione peggiore ovvero che fuori abbiano la mia tabella di partenza. Tutta sta cosa di positivo mi migliora l'utilità perché sennò avrei dovuto cancellare altri dati o generalizzare di più, mi complica il problema e mi richiede di guardare la tabella privata per prendere una decisione di rilascio, però non me lo complica a tal punto da richiedere conoscenza che il proprietario dei dati non ha per sapere se il rilascio va bene o no.



Cosa non va bene? Il fatto che partendo dalle 4 tabella sotto, qualsiasi tupla si matcha ad almeno 2 sopra non vale, per la quinta tupla c'è un match esatto, mentre nella 5 l'ultimo asian è per forza l'ultimo della prima tabella dato che gli altri 94141 sono già presi , quindi quell'asian rosso della 5 tabella deve essere per forza un 94142.

Background Knowledge

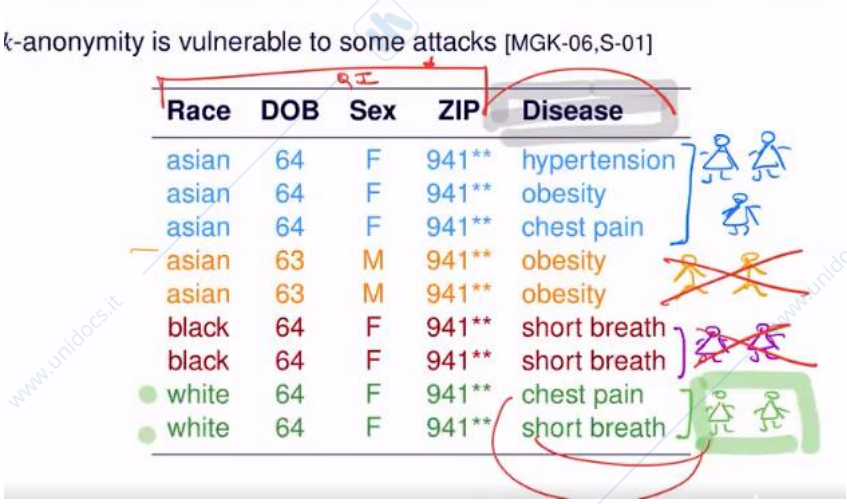
Mi dice: per k anonymity ti sei preoccupato del problema del linking tramite i quasi identifier, con l'assunzione che la gente può prendere tabelle da altre parti e andare a linkare, però ci potrebbe essere altra conoscenza che qualcuno potrebbe avere, ad esempio so che Hellen è nella tabella perché sono un vicino di casa, se io so i suoi dati so che è una delle due, se io so che hellen corre 2 ore al giorno di certo non ha il fiato corto, ma sicuramente per esclusione ha dolore al petto.



Race	DOB	Sex	ZIP	Disease
...	...	...	...	...
white	64	F	941**	chest pain
white	64	F	941**	short breath

Quindi l'osservatore può avere conoscenze a livello di istanza e scartare delle possibilità rispetto alla persona e quindi inferire qualcosa, da un'incertezza di 2 quindi sono arrivato ad un'incertezza di 1.

l-diversity mi dice: sempre costruendo su k anonimity, quando tu fai i blocchetti di tuple per cui vuoi almeno k persone dentro, chiedi che siano l-diversi, ovvero che abbiano almeno l valori ben rappresentati, l'idea è che ti proteggo rispetto ad attacchi di omogeneità e se ho almeno l valori quello che succede è che se mi fai ragionamenti di questo tipo ad esempio l va a correre per cui non può avere il fiato corto, di queste cose qua devi farmi almeno l-1 assunzioni di questo tipo. Inoltre il problema della background knowledge è più difficile. Esempio:



k-anonymity is vulnerable to some attacks [MGK-06,S-01]

Race	DOB	Sex	ZIP	Disease
asian	64	F	941**	hypertension
asian	64	F	941**	obesity
asian	64	F	941**	chest pain
asian	63	M	941**	obesity
asian	63	M	941**	obesity
black	64	F	941**	short breath
black	64	F	941**	short breath
white	64	F	941**	chest pain
white	64	F	941**	short breath

Il gruppo blu è 3-diverso, il verde è 2-diverso, l'arancio è 1-diverso, il rosso è 1-diverso, per esclusione è più difficile dire se Hellen ha il fiato corto se ci sono altre malattie.

Skewness attack

L'attacco di asimmetria (Skewness Attack) si verifica quando la distribuzione blocco q è diversa dalla distribuzione nella popolazione originale. Nell'esempio sottostante il 20% della popolazione soffre di diabete; Il 75% delle tuple in un blocco Q ha il diabete, da questo si può dedurre che le persone rappresentate nel blocco q hanno maggiori probabilità di soffrire di diabete

Race	DOB	Sex	ZIP	Disease
black	64	F	941**	diabetes
black	64	F	941**	short breath
black	64	F	941**	diabetes
black	64	F	941**	diabetes

Similarity attack

Tu mi chiedi che il gruppo sia l diverso, ovvero che ci siano almeno l malattie diverse, però stai attento a cosa vuol dire diverso, se tu mi metti in un gruppo in cui ci sono o l'ulcera allo stomaco o la gastrite in realtà agli occhi dell'osservatore io ho un'l diversità di 2, però l'osservatore sa più o meno di cosa sono ammalato

Race	DOB	Sex	ZIP	Disease
black	64	F	941**	stomach ulcer
black	64	F	941**	stomach ulcer
black	64	F	941**	gastritis

K=3

l=2

Quindi se le malattie sono diverse ma sono molto vicine e sono della stessa famiglia in realtà magari il mio paziente non è tanto protetto, perché qualcuno non saprà la malattia precisa ma più o meno in realtà la sa.

t-closeness

Mi dice: tu fai pure k anonimity, fai pure l diversity, però non accontentarti, quello che dovresti cercare di fare quando fai i tuoi blocchi devi guardare che abbiano k valori per i quasi identifier, devi guardare che siano almeno l diversity, cerca anche di vedere che la distribuzione dei valori sensibili nel gruppetto non sia troppo diversa da quella che è la distribuzione nel mondo reale, vuol dire che se io sono un ospedale che rilascia i dati dovrei assicurarmi che se io faccio k anon che i miei gruppetti abbiano una distribuzione delle malattie simile a quella che c'è nel mondo reale. Se nel mondo reale ad esempio il 20% ha il diabete il gruppetto qui sotto non mi piacerà, una tupla su 4 può avere il diabete, 3 su 4 non mi va bene

Race	DOB	Sex	ZIP	Disease
black	64	F	941**	diabetes
black	64	F	941**	short breath
black	64	F	941**	diabetes
black	64	F	941**	diabetes

Quindi t closeness può essere applicato dopo aver fatto k anon e anche l diversity.

Rilasci multipli (o longitudinali)

Ad esempio un ospedale che pubblica i dati ogni settimana, ho rilasci multipli che si possono riferire allo stesso dataset (in cui le persone sono sempre le stesse) o dataset diversi (es quando in un reparto c'è gente che entra e gente che esce). Esempio

T_1

DOB	Sex	ZIP	Disease
74	*	941**	aids
74	*	941**	flu
74	*	941**	flu
74	*	941**	aids
64	*	941**	flu
64	*	941**	short breath
64	*	941**	flu
64	*	941**	aids

4-anonymized table at time t_1

T_2

DOB	Sex	ZIP	Disease
[70-80]	F	9414*	hypertension
[70-80]	F	9414*	gastritis
[70-80]	F	9414*	aids
[70-80]	F	9414*	gastritis
[60-70]	M	9413*	flu
[60-70]	M	9413*	aids
[60-70]	M	9413*	flu
[60-70]	M	9413*	gastritis

4-anonymized table at time t_2

An adversary knows that Alice, born in 1974 and living in area 94142, is in both releases

Cosa possiamo dire su alice? Che ha l'AIDS.

Questo attacco prende il nome di attacco di intersezione su dati longitudinali.

T_1

DOB	Sex	ZIP	Disease
74	*	941**	aids
74	*	941**	flu
74	*	941**	flu
74	*	941**	aids
64	*	941**	flu
64	*	941**	short breath
64	*	941**	flu
64	*	941**	aids

4-anonymized table at time t_1

T_2

DOB	Sex	ZIP	Disease
[70-80]	F	9414*	hypertension
[70-80]	F	9414*	gastritis
[70-80]	F	9414*	aids
[70-80]	F	9414*	gastritis
[60-70]	M	9413*	flu
[60-70]	M	9413*	aids
[60-70]	M	9413*	flu
[60-70]	M	9413*	gastritis

4-anonymized table at time t_2

An adversary knows that Frank, born in 1964 and living in area 94132, is in T_1 but not in T_2

Esempio simile qui, ho i rilasci di 2 settimane diverse, posso dire che frank soffre di fiato corto.

Per fare inferenza abbiamo sfruttato il fatto che una malattia era unica nella prima tabella e poi non c'era più, **m-invariance** mi dice che se il rilascio non è unico ma hai una storia di rilasci quello che devi fare è stare attento ai valori sensibili, mi chiede di avere al più un valore sensibile per ogni gruppetto e che i valori sensibili in cisascun gruppetto siano sempre gli stessi, es

T_1

DOB	Sex	ZIP	Disease
64	*	941**	flu
64	*	941**	short breath
64	*	941**	flu
64	*	941**	aids

4-anonymized table at time t_1

T_2

DOB	Sex	ZIP	Disease
[60-70]	M	9413*	flu
[60-70]	M	9413*	aids
[60-70]	M	9413*	flu
[60-70]	M	9413*	gastritis

4-anonymized table at time t_2

Non gli piace che prima c'era short breath e poi gastritis, vorrebbe avere short breath anche dopo, questo vuol dire fare i gruppi in modo diverso, così come non gli piace che prima c'era una distribuzione diversa di valori.

Se ti impongo che la distribuzione di valori sia sempre quella allora non si riesce più a fare attacchi di intersezione di questo tipo. …….. incompleto

Per fare fronte a questi rilasci longitudinali si può utilizzare la m-invariance: se ci sono stati dei rilasci degli stessi dati in passato è necessario mantenere per i successivi le linee guida mediante le quali essi sono stati effettuati (ad esempio mantenendo le stesse generalizzazioni). Una sequenza soddisfa m-invariance se:

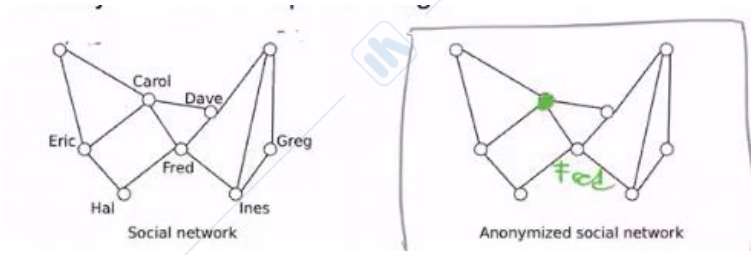
• ciascuna classe di equivalenza (gruppo) ha almeno m tuple;

• nessun valore sensibile appare più di una volta in ogni classe di equivalenza (son tutti diversi!);

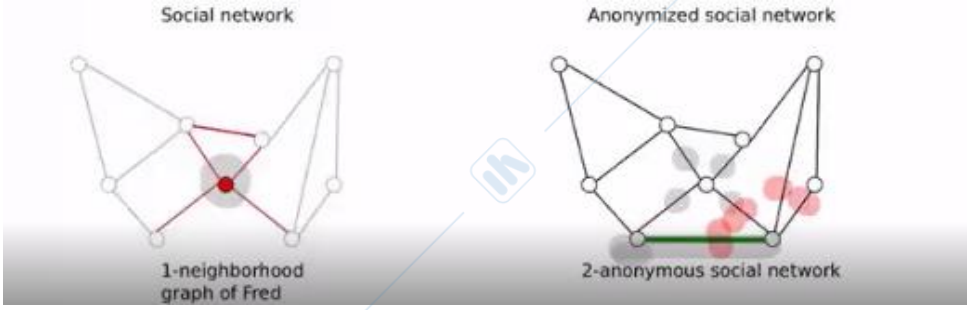
• per ogni tupla le classi di equivalenza in cui t è presente hanno gli stessi valori sensibili (ad esempio i gruppi devono avere le stesse malattie nelle due settimane in cui vengono rilasciati i dati).

k-anonimity in social networks

Vuol dire che tu hai una rete sociale, quindi gente collegata ad altra gente, la pubblichi in modo anonimo e ottieni un grafo di rete sociale anonimo. Il problema di proteggere l'identità di questa cosa si declina nel fatto che qualcuno riesca a guardare la tua versione anonima dei dati e riesca a dire ad esempio che un vertice è una persona in particolare.



In una rete sociale anonima se ho un nodo che ha grado 1 e un nodo che ha grado 1 milione, posso capire chi sono le persone, se quello che mi distingue è il grado e voglio fare k anonymity sul grado, allora ho bisogno di almeno k nodi con le stesse caratteristiche e si può fare in due modi: o aggiungendo o cancellando archi. In questo caso aggiungo archi



Il nodo rosso è Fred, per confonderlo aggiungo un altro arco in modo tale che non è l'unico ad avere 4 archi ma anche altre persone, così confondo la gente.

Neighborhood attack