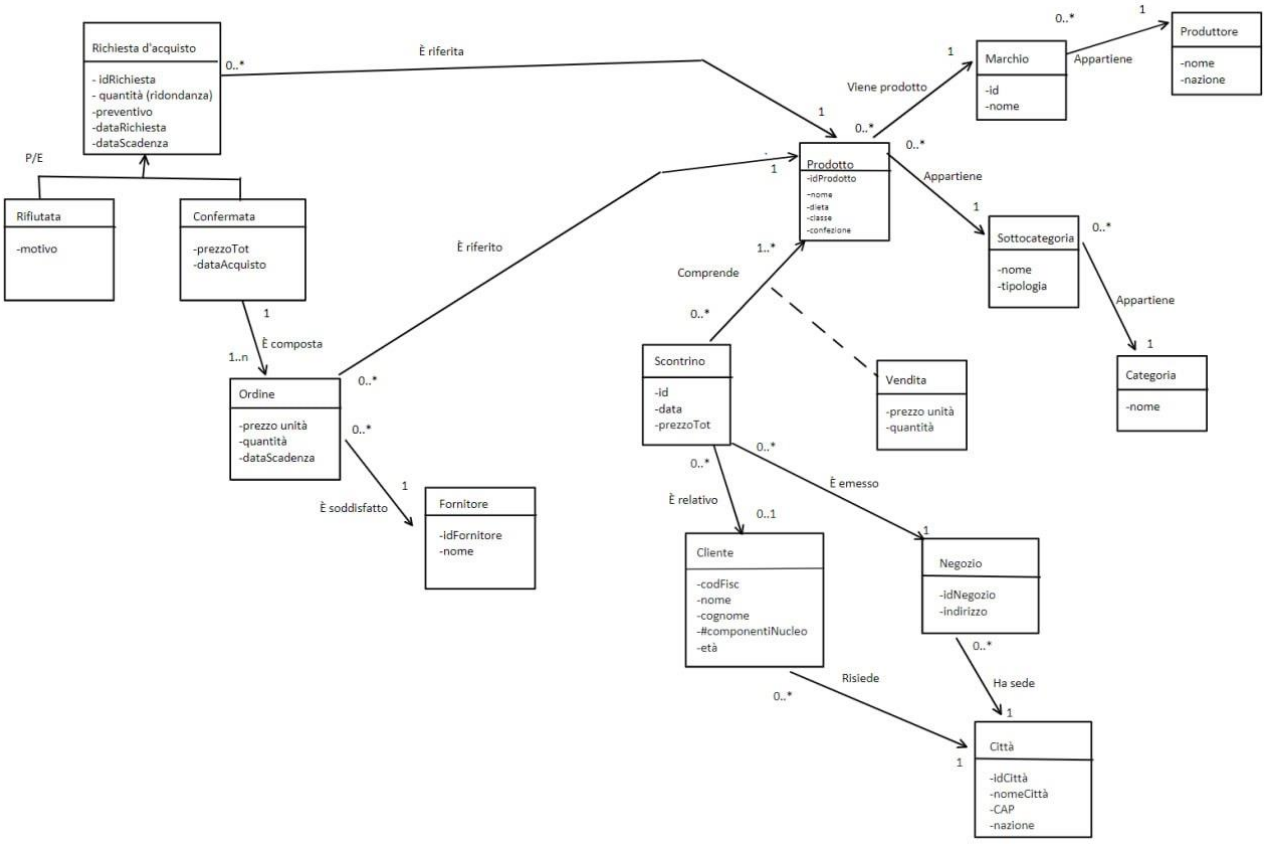
Progetto Sistemi Informativi

Sommario

UML – Class Diagram

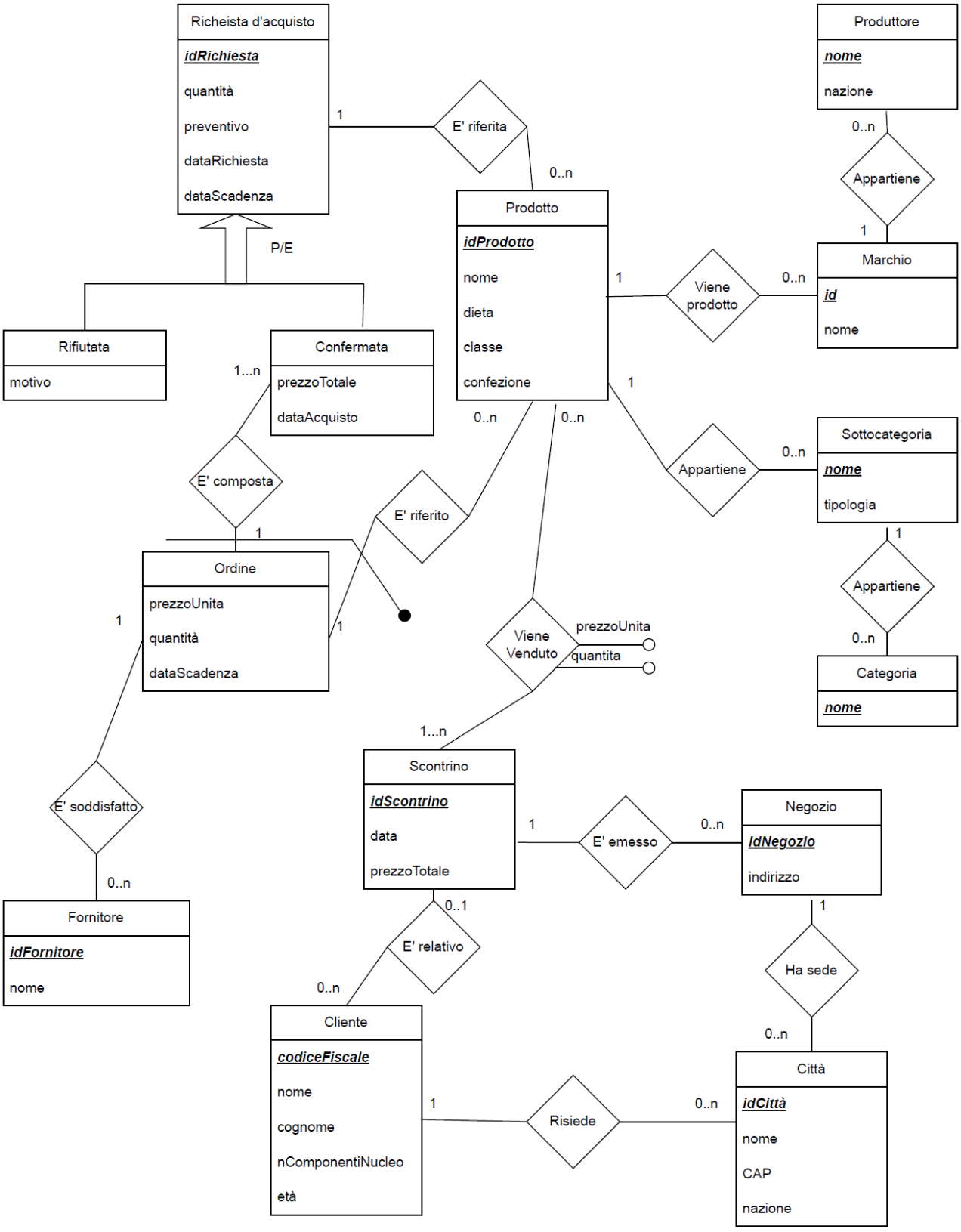
Lo schema UML è relativo al DB operativo del gestore del magazzino e dell'ufficio acquisti. Attraverso quest'ultimo modelliamo concettualmente i dati; lo schema risultante è molto essenziale e con pochi attributi descrittivi. Analisi dei casi più importanti:

- Ordine: Rappresenta un ordine che viene effettuato in seguito alla conferma di una richiesta d'acquisto. Esso è relativo ad una specifica richiesta d'acquisto, ad uno specifico fornitore attraverso il quale viene effettuato e ad un solo prodotto così da poterne indicare la quantità e il prezzo.
- Cliente: Memorizziamo i suoi solo nel momento in cui accetta la campagna di fidelizzazione, dandoci il consenso e alcuni dei suoi dati
- Vendita: è un association class fra prodotto e scontrino; ci permette di definire alcuni attributi importanti riguardo all'atto di vendita
- Richiesta d'acquisto: è il passo precedente all'atto di acquisto (la sua conferma o rifiuto è definita nel modello BPMN); richiesta d'acquisto è la generalizzazione di due sue specializzazioni "rifiutata" e "confermata"



Entity Relationship

Traduciamo l'UML in uno schema concettuale: le classi diventano entità, le associazioni diventano relazioni (invertendo le cardinalità), l'association class è tradotta come una relazione, gli attributi dell'association class come attributi di relazione



Schema Relazionale

RICHIESTA_ACQUISTO(idRichiesta, quantità, dataRichiesta, dataScadenzaRichiesta, idProdotto : PRODOTTO)

RICHIESTA_ACQUISTO_RIFIUTATA(idRichiesta : RICHIESTA, motivoRifiuto)

RICHIESTA_ACQUISTO_CONFERMATA(idRichiesta : RICHIESTA, prezzoTot, dataAcquisto)

ORDINE(idRichiesta : RICHIESTA_ACQUISTO_CONFERMATA, idFornitore: FORNITORE, prezzoUnità, quantità, dataScadenza, idProdotto : PRODOTTO)

FORNITORE(idFornitore, nome)

PRODOTTO(idProdotto, nome, confezione, dieta(nullable), classe(nullable), marchio: MARCHIO, nomeSotCat: SOTTOCATEGORIA)

MARCHIO(idMarchio, nome, produttore: PRODUTTORE)

PRODUTTORE(idProduttore, nome, nazione)

CATEGORIA(nomeCat)

SOTTOCATEGORIA(nomeSotCat, tipologia(nullable), nomeCat: CATEGORIA)

VENDITA(idScontrino: SCONTRINO, idProdotto: Prodotto, quantità, prezzoUnità)

SCONTRINO(idScontrino, dataVendita, prezzoTot, cliente: CLIENTE(nullable), idNegozio: NEGOZIO)

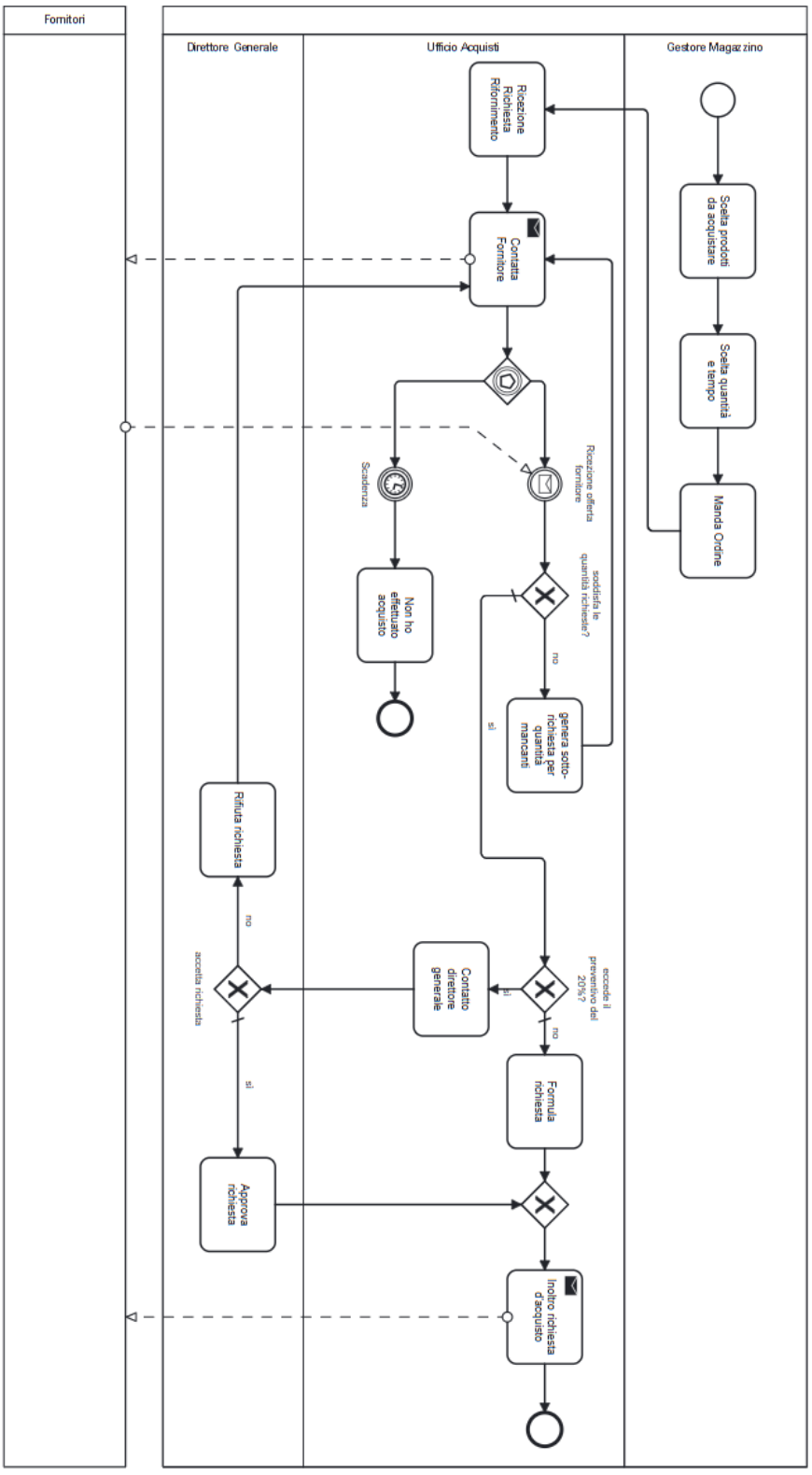
NEGOZIO(idNegozio, indirizzo, idCittà: CITTA')

CITTA'(idCittà, nomeCittà, CAP, regione, nazione)

CLIENTE(codFisc, nome, cognome, numComponentiNucleo, idCittà: CITTA')

BPMN

Implementiamo il processo di acquisizione dei prodotti in magazzino. Il pool principale (quello dell'azienda) è composto da 3 Lanes: gestore magazzino, ufficio acquisti e direttore generale. Il processo segue il paradigma CRASO: in questo caso il cliente è l'azienda stessa che attende le risposte dai fornitori, l'insieme delle attività serve a determinare se una richiesta del direttore generale è soddisfacibile o meno. Il processo è categorizzabile come *Orchestrazione* in quanto descrive un processo all'interno di una singola entità aziendale che è contenuta all'interno di un unico Pool

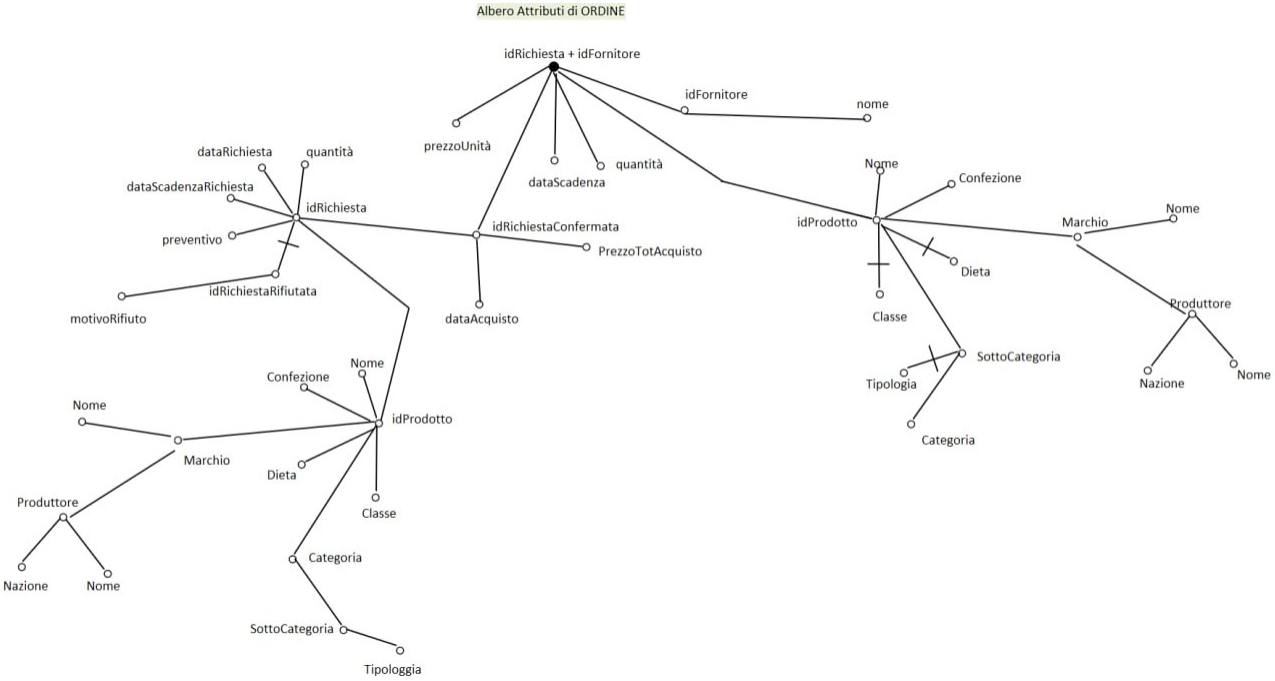


DFM

Per creare il design di una Data Warehouse è necessario implementare il DFM, un linguaggio grafico per la modellazione concettuale.
Nel creare il DFM ci siamo focalizzati su 2 fatti di interesse: Ordine e Vendita. Questi rappresentano le due attività principali dell'impresa e gestiscono rispettivamente la logistica di entrata e di uscita. Sono rappresentati da tabelle il cui contenuto varia nel tempo e sono le due che vengono aggiornate più frequentemente: possiamo quindi dire inseguano il presente. Inoltre presentano entrambe almeno due chiavi esterne e sono quindi a cavallo tra più relazioni. Una volta individuati i fatti abbiamo creato l'albero degli attributi per ciascun fatto, come mostrato dalle seguenti figure.

Ordine

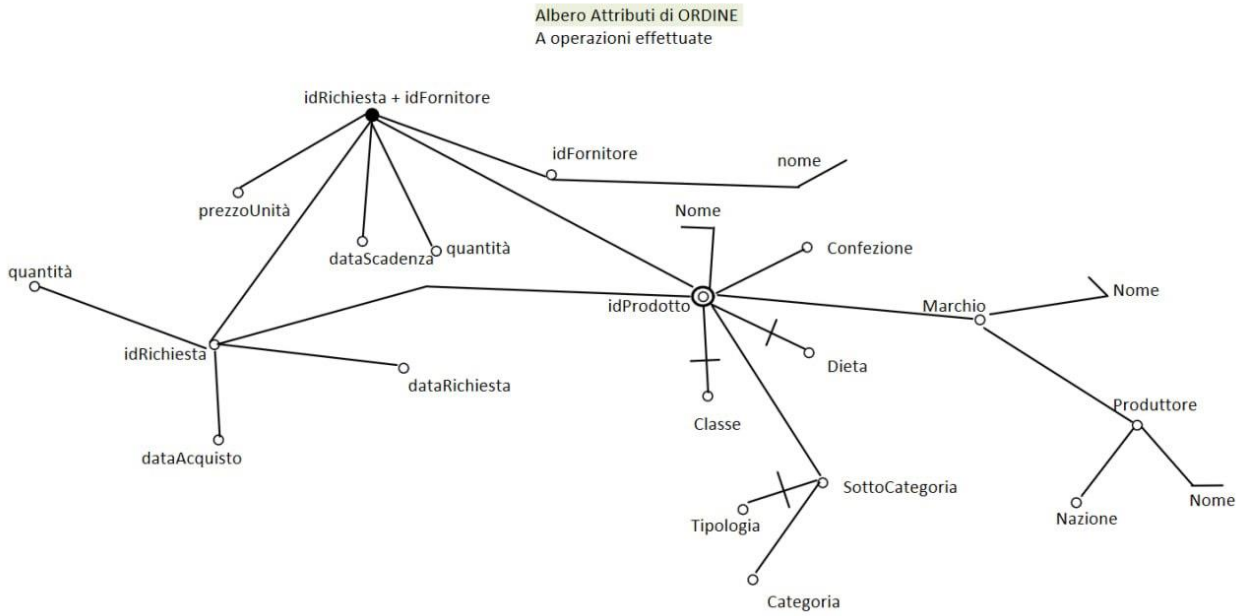
ORDINE – Albero Attributi



ORDINE – OPERAZIONI EFFETTUATE

OPERAZIONI EFFETTUATE sull'albero degli attributi di ORDINE:

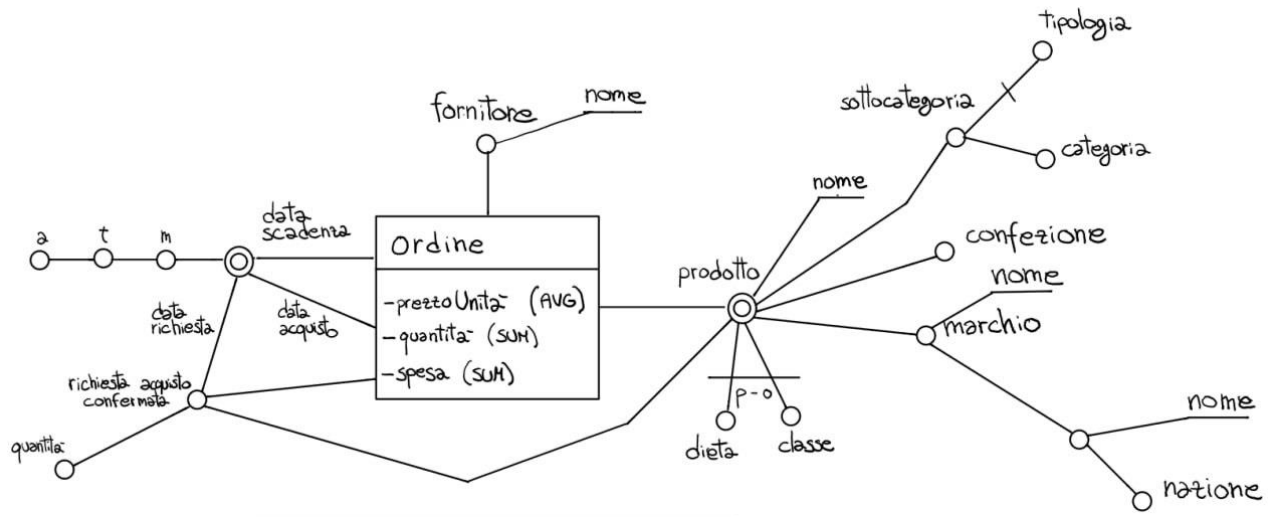
1. Pruning su *idRichiestaRifiutata* in quanto l'algoritmo non rileva che è opzionale, anzi, dal momento che il fatto è ORDINE e la gerarchia della richiesta d'acquisto è di tipo esclusiva, l'ordine non può corrispondere ad una richiesta rifiutata.
2. Grafting su *idRichiesta* di RICHIESTA_ACQUISTO dal momento che RICHIESTA_ACQUISTO e RICHIESTA_ACQUISTO_CONFERMATA si trovano ora in relazione 1 a 1.
3. Rimozione attributo dataScadenzaRichiesta in quanto non è rilevante non considerando più le richieste d'acquisto rifiutate.
4. Creazione gerarchia condivisa per il prodotto.
5. Rimozione attributo preventivo non necessario.
6. Rimozione attributo PrezzoTotAcquisto: andrebbe a costituire un valore continuo da discretizzare non necessario.
7. Nome del fornitore come attributo descrittivo.
8. Nome del prodotto come attributo descrittivo.
9. Nome del marchio come attributo descrittivo.
10. Nome del produttore come attributo descrittivo



ORDINE – MODIFICHE EFFETTUATE

MODIFICHE EFFETTUATE nella realizzazione del DFM di ORDINE:

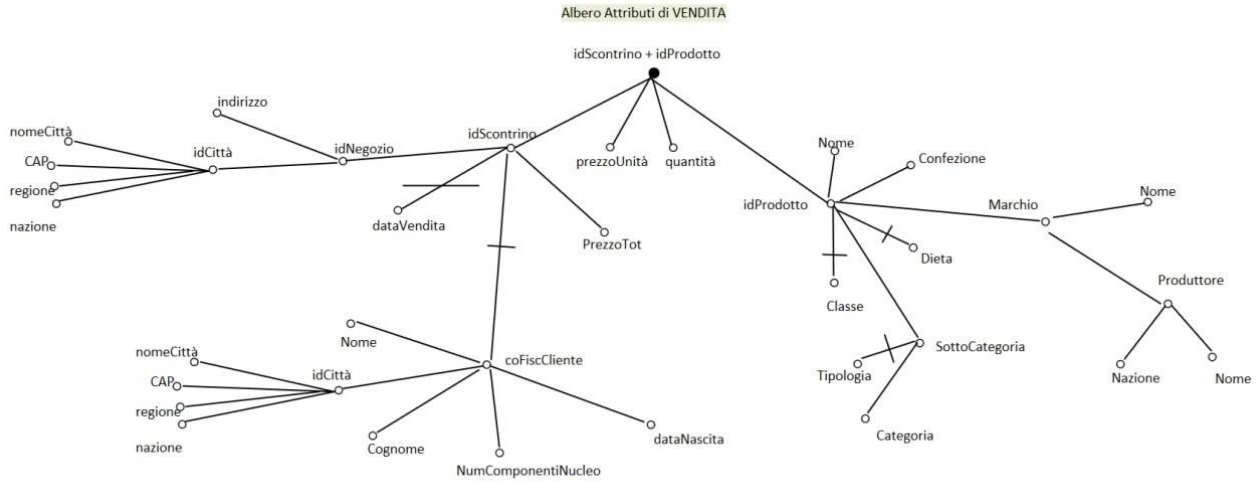
1. Aggiunta nel DFM una gerarchia di rollup a dataAcquisto per ottenere una vista aggregata dei dati degli acquisti.
2. Creazione gerarchia condivisa con dataAcquisto, dataScadenza e dataRichiesta.
3. Aggiunta relazione di coverage tra gli attributi opzionali dieta e classe di prodotto.
4. Coverage di tipo parziale in quanto dieta e classe possono anche non essere valorizzati entrambi per uno stesso prodotto e overlapping in quanto possono anche essere valorizzati entrambi per lo stesso prodotto
5. Individuazione misure: prezzoUnità, quantità e spesa (derivata, non rappresentata a livello operativo).



ORDINE	Prodotto	Fornitore	dataScadenza	Richiesta d'acquisto conf
prezzoUnità (misura unitaria)	AVG	AVG	AVG	AVG
Quantità (misura di flusso)	SUM	SUM	SUM	SUM
Spesa (misura derivata)	SUM	SUM	SUM	SUM

Vendita

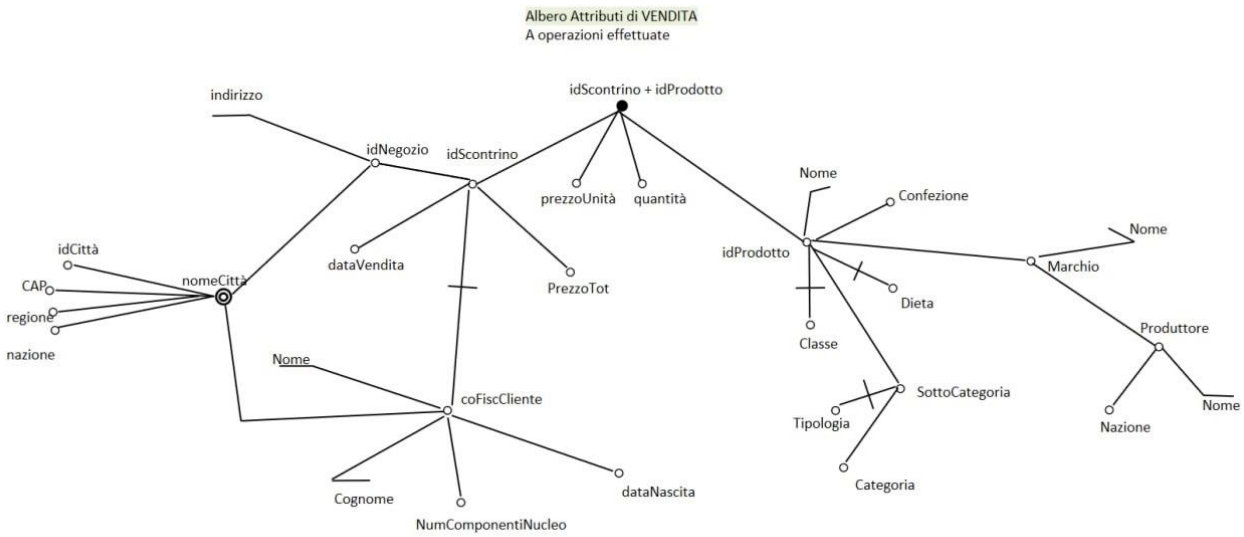
VENDITA – Albero Attributi



VENDITA – OPERAZIONI EFFETTUATE

OPERAZIONI EFFETTUATE sull'albero degli attributi di VENDITA

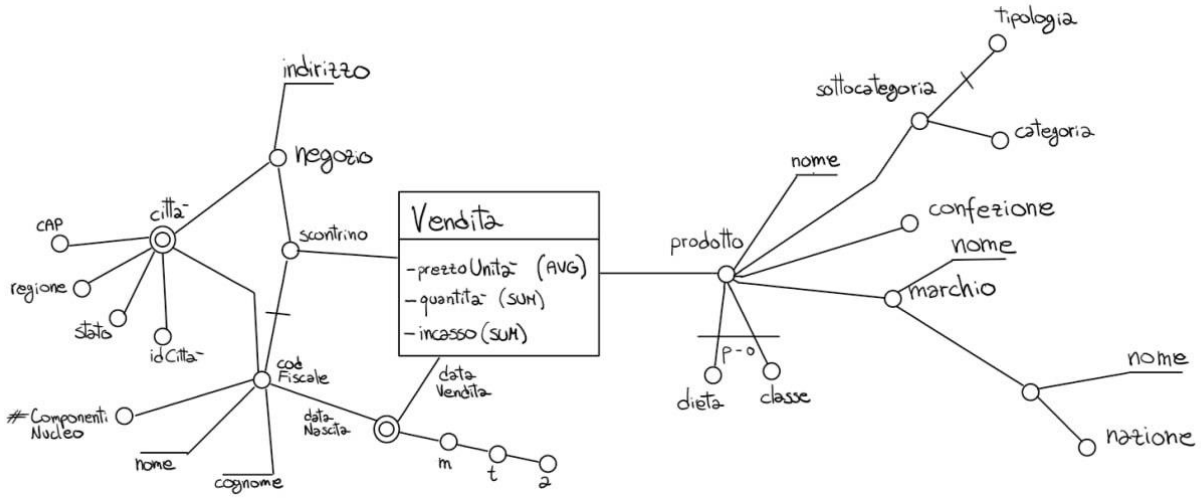
1. Inserimento gerarchia condivisa per la città.
2. Operazione di "taglia e cuci" su idCittà e nomeCittà poiché aggregheremo più di sovente sul nome.
3. Nome e cognome del cliente come attributi descrittivi.
4. Nome del prodotto come attributo descrittivo.
5. Nome del marchio come attributo descrittivo.
6. Nome del produttore come attributo descrittivo
7. Indirizzo del negozio come attributi descrittivo poiché difficilmente utilizzeremo per aggregazioni.



VENDITA – MODIFICHE EFFETTUATE

MODIFICHE EFFETTUATE nella realizzazione del DFM di VENDITA:

1. Aggiunta nel DFM di una gerarchia di rollup a dataVendita dello scontrino per ottenere una vista aggregata dei dati delle vendite.
2. Inserimento di una gerarchia condivisa con la dataNascita di un cliente.
3. Inserimento di una misura 'incasso' non rappresentata a livello operativo, dal momento che è preferibile fare le operazioni di aggregazione e di calcolo a priori.



VENDITA	Prodotto	Scontrino
prezzoUnità (misura unitaria)	AVG	AVG
Quantità (misura di flusso)	SUM	SUM
Incasso (misura derivata)	SUM	SUM

Star Schema

Gestione cambiamenti dimensioni estensionali

1. Cambia il nome di un fornitore:

Dal momento che i fornitori verranno analizzati di frequente per varie valutazioni, se un fornitore cambia nome potrebbe voler indicare una nuova gestione e quindi potrebbe esserci utile mantenere le informazioni storiche separate (ad esempio per vedere se vi è stato un aumento generale dei prezzi con la nuova gestione)

Per questo lo interpretiamo come scenario 'Oggi o ieri' e scegliamo di applicare una gerarchia dinamica di tipo 3.

2. Cambia il nome di un produttore o di un marchio.

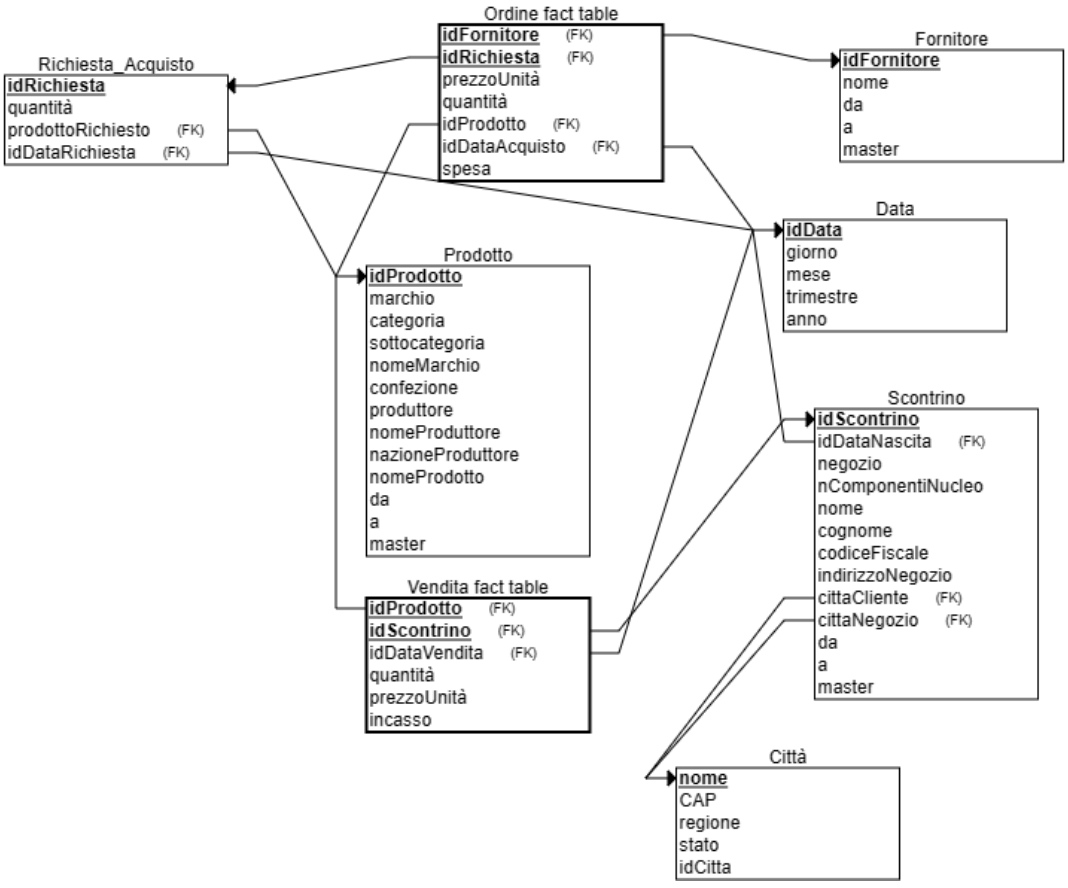
L'azienda non deve solo valutare da chi comprare ma anche quali prodotti comprare. Per le stesse ragioni sopra elencate, potrei essere interessato ad analizzare i guadagni dovuti alla vendita di determinati prodotti inseguito al cambio di nome di un marchio/produttore e di conseguenza scegliere di acquisirne in quantità maggiori o rifornirmi con prodotti di un altro marchio.

Per queste ragioni lo interpretiamo scenario 'Oggi o ieri' e scegliamo di applicare una gerarchia dinamica di tipo 3.

3. Cambio l'indirizzo di un negozio.

Per ragioni di analisi delle vendite di un negozio è fondamentale distinguere le vendite prima e dopo un trasferimento. Per questo utilizziamo lo scenario 'Oggi o ieri' e applichiamo una gerarchia di tipo 2.

Quindi quando un negozio cambia indirizzo lo consideriamo come un nuovo negozio.



Query

Query 1: Andamento dei ricavi per categoria con granularità al trimestre

```
SELECT p.categoria, d.anno, d.Trimestre, SUM(v.quantita * v.prezzoUnita) AS Ricavi

FROM venditaft v JOIN datadt AS d ON d.idData = v.idDataVendita
        JOIN Prodottodt AS p ON p.IdProdotto = v.idProdotto

GROUP BY p.categoria,d.anno, ROLLUP (d.trimestre)
```

Esempio risultato query

	categoria character varying	anno integer	trimestre integer	ricavi bigint
1	liquidi	2022	1	7000
2	liquidi	2022	3	45000
3	liquidi	2022	4	200000
4	liquidi	2022	[null]	252000
5	solidi	2022	1	70000
6	solidi	2022	2	22500
7	solidi	2022	3	30000
8	solidi	2022	[null]	122500

Query 2: Costi (prezzo di acquisto) per fornitore negli ultimi tre anni

```
WITH Costi2020 AS(

        SELECT f.master_key, f.nome, o.spesa AS costoOrdine2020, p.categoria, p.sottocategoria,
        p.nome_prodotto AS prodotto

        FROM ordineft as o JOIN fornitoredt as f ON o.idFornitore = f.idFornitore JOIN prodottodt as p ON
        p.idProdotto = o.idProdotto JOIN richiestaacquistodt as r ON r.idRichiesta = o.idRichiesta JOIN
        datadt as d ON d.idData = o.idDataAcquisto

        WHERE d.anno = 2020 ),

Costi2021 AS(

        SELECT f.master_key, f.nome, o.spesa AS costoOrdine2021, p.categoria, p.sottocategoria,
        p.nome_prodotto AS prodotto

        FROM ordineft as o JOIN fornitoredt as f ON o.idFornitore = f.idFornitore JOIN prodottodt as p ON
        p.idProdotto = o.idProdotto JOIN richiestaacquistodt as r ON r.idRichiesta = o.idRichiesta JOIN
        datadt as d ON d.idData = o.iddataAcquisto

        WHERE d.anno = 2021 ),

Costi2022 AS(

        SELECT f.master_key, f.nome, p.categoria, p.sottocategoria, p.nome_prodotto AS prodotto, o.spesa
        AS costoOrdine2022

        FROM ordineft as o JOIN fornitoredt as f ON (o.idFornitore = f.idFornitore) JOIN prodottodt as p ON
        (p.idProdotto = o.idProdotto) JOIN richiestaacquistodt as r ON (r.idRichiesta = o.idRichiesta) JOIN
        datadt as d ON (d.idData = o.iddataAcquisto)

        WHERE d.anno = 2022 )
```

```
SELECT COALESCE(c20.master_key,c21.master_key,c22.master_key) AS master,
        COALESCE(c20.nome,c21.nome,c22.nome) AS fornitore,
        COALESCE(c20.categoria,c21.categoria,c22.categoria) AS categoria_prodotto,
        COALESCE(c20.sottocategoria,c21.sottocategoria,c22.sottocategoria) AS sottocategoria_prodotto,
        COALESCE(c20.prodotto,c21.prodotto,c22.prodotto) AS nome_prodotto,
        COALESCE(sum(c20.costoOrdine2020),0) AS costo2020,
        COALESCE(sum(c21.costoOrdine2021),0) AS costo2021,
        COALESCE(sum(c22.costoOrdine2022),0) AS costo2022

FROM Costi2020 c20
        FULL JOIN Costi2021 c21 ON (c20.nome = c21.nome AND c20.categoria = c21.categoria AND
        c20.sottocategoria = c21.sottocategoria AND c20.prodotto = c21.prodotto)
        FULL JOIN Costi2022 c22 ON (c20.nome = c22.nome AND c20.categoria = c22.categoria AND
        c20.sottocategoria = c22.sottocategoria AND c20.prodotto = c22.prodotto)

GROUP BY master,fornitore,ROLLUP(categoria_prodotto,sottocategoria_prodotto,nome_prodotto)
```

Esempio risultato query

	master integer	fornitore character varying	categoria_prodotto character varying	sottocategoria_prodotto character varying	nome_prodotto character varying	costo2020 bigint	costo2021 bigint	costo2022 bigint
1	1	papero	liquidi	bibite_gasate	kola	0	0	304000
2	1	papero	liquidi	bibite_gasate	[null]	0	0	304000
3	1	papero	liquidi	[null]	[null]	0	0	304000
4	1	papero	[null]	[null]	[null]	0	0	304000
5	1	pippo	liquidi	bibite_gasate	kola	0	0	352000
6	1	pippo	liquidi	bibite_gasate	[null]	0	0	352000
7	1	pippo	liquidi	[null]	[null]	0	0	352000
8	1	pippo	solidi	creme	nutellaPRO	0	55000	0
9	1	pippo	solidi	creme	[null]	0	55000	0
10	1	pippo	solidi	[null]	[null]	0	55000	0
11	1	pippo	[null]	[null]	[null]	0	55000	352000
12	2	marco	liquidi	bibite_gasate	kolaZERO	0	0	145000
13	2	marco	liquidi	bibite_gasate	[null]	0	0	145000
14	2	marco	liquidi	[null]	[null]	0	0	145000
15	2	marco	[null]	[null]	[null]	0	0	145000
16	2	pluto	liquidi	bibite_gasate	kolaZERO	0	0	144000

Query 3: Giacenze vs quantità vendute per prodotto negli ultimi 4 trimestri

```
WITH Ordinati as(

        SELECT idprodotto,
        SUM(CASE WHEN trimestre <= 1 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaOrdinateT1,
        SUM(CASE WHEN trimestre <= 2 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaOrdinateT2,
        SUM(CASE WHEN trimestre <= 3 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaOrdinateT3,
        SUM(CASE WHEN trimestre <= 4 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaOrdinateT4

        FROM ordineft
        JOIN datadt ON iddata = iddataacquisto

        GROUP BY idprodotto

),

------------------------- VENDITE -------------------------------

Vendute as(

        SELECT idprodotto,
        SUM(CASE WHEN trimestre <= 1 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaVenduteT1,
        SUM(CASE WHEN trimestre <= 2 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaVenduteT2,
        SUM(CASE WHEN trimestre <= 3 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaVenduteT3,
        SUM(CASE WHEN trimestre <= 4 AND anno = 2022 OR (anno < 2022) THEN quantita ELSE 0 END) as
        quantitaVenduteT4

        FROM venditaft
        JOIN datadt ON iddata = iddatavendita

        GROUP BY idprodotto

),

------------------------- GIACENZE -------------------------------

Giacenze AS(

SELECT Ordinati.idprodotto,

        (quantitaOrdinateT1 - COALESCE(quantitaVenduteT1,0)) AS giacenzaT1,
        (quantitaOrdinateT2 - COALESCE(quantitaVenduteT2,0)) AS giacenzaT2,
        (quantitaOrdinateT3 - COALESCE(quantitaVenduteT3,0)) AS giacenzaT3,
        (quantitaOrdinateT4 - COALESCE(quantitaVenduteT4,0)) AS giacenzaT4

        FROM Ordinati LEFT JOIN Vendute ON Ordinati.idprodotto = Vendute.idprodotto

),
```

```
VenditeTrimestri AS(

        SELECT venditaft.idprodotto,
        SUM(CASE WHEN trimestre = 1 AND anno = 2022 THEN quantita ELSE 0 END) AS VenditeT1,
        SUM(CASE WHEN trimestre = 2 AND anno = 2022 THEN quantita ELSE 0 END) AS VenditeT2,
        SUM(CASE WHEN trimestre = 3 AND anno = 2022 THEN quantita ELSE 0 END) AS VenditeT3,
        SUM(CASE WHEN trimestre = 4 AND anno = 2022 THEN quantita ELSE 0 END) AS VenditeT4

        FROM venditaft JOIN prodottodt ON (venditaft.idProdotto = prodottodt.idProdotto)
        JOIN datadt ON (iddata = iddatavendita)

        GROUP BY (venditaft.idprodotto)

)

SELECT COALESCE(Prodottodt.categoria, 'TOTALE') AS categoria,
        COALESCE(Prodottodt.sottocategoria, 'TOTALE CATEGORIA') AS sottocategoria,
        COALESCE(Prodottodt.nome_prodotto, 'TOTALE SOTTOCATEGORIA') AS nome_prodotto,
        SUM(giacenzaT1) AS giacenzaT1, SUM(VenditeT1) AS VenditeT1, SUM(giacenzaT2) AS giacenzaT2,
        SUM(VenditeT2) AS VenditeT2, SUM(giacenzaT3) AS giacenzaT3, SUM(VenditeT3) AS VenditeT3,
        SUM(giacenzaT4) AS giacenzaT4, SUM(VenditeT4) AS VenditeT4

FROM prodottodt
        RIGHT JOIN Giacenze AS g ON g.idprodotto = prodottodt.idprodotto
        FULL OUTER JOIN VenditeTrimestri ON g.idprodotto = VenditeTrimestri.idprodotto


GROUP BY ROLLUP (Prodottodt.categoria, Prodottodt.sottocategoria, Prodottodt.nome_prodotto)
```

Esempio risultato query

	categoria character varying	sottocategoria character varying	nome_prodotto character varying	giacenzat1 numeric	venditet1 numeric	giacenzat2 numeric	venditet2 numeric	giacenzat3 numeric	venditet3 numeric	giacenzat4 numeric	venditet4 numeric
1	liquidi	bibite_gasate	kola	2650	350	2650	0	2550	100	2350	200
2	liquidi	bibite_gasate	kolaZERO	290	0	650	0	650	0	450	200
3	liquidi	bibite_gasate	TOTALE SOTTOCATEGORIA	2940	350	3300	0	3200	100	2800	400
4	liquidi	TOTALE CATEGORIA	TOTALE SOTTOCATEGORIA	2940	350	3300	0	3200	100	2800	400
5	solidi	creme	nutella	360	100	360	0	360	0	360	0
6	solidi	creme	nutellaPRO	550	0	400	150	200	200	200	0
7	solidi	creme	TOTALE SOTTOCATEGORIA	910	100	760	150	560	200	560	0
8	solidi	TOTALE CATEGORIA	TOTALE SOTTOCATEGORIA	910	100	760	150	560	200	560	0
9	TOTALE	TOTALE CATEGORIA	TOTALE SOTTOCATEGORIA	3850	450	4060	150	3760	300	3360	400