

# Analitikus adatfeldolgozás

Adattárház

Adatkocka

Adatbányászat

# Áttekintés

- ◆ A hagyományos adatbázisokat sok, apró, egyszerű lekérdezésre hangolták
- ◆ A jelenlegi alkalmazások kevesebb, de idő igényesebb, bonyolultabb lekérdezéseket használnak
- ◆ Ezért modernebb adatarchitektúrákat fejlesztettek ki azért, hogy a bonyolultabb „analitikus”, elemző jellegű lekérdezéseket kezelni tudják

# Adattárház

- ◆ Jelenleg ez az egyik legelterjedtebb formája az adatintegrációnak.
  - ◆ Egyetlen egy közös adatbázisba másolják (adattárház) az adatokat és napra készen tartják.
  - ◆ Módszere: periodikus aktualizálás, gyakran éjszaka.
  - ◆ Gyakran az analitikus lekérdezések végett hozzák létre.

# OLTP

- ◆ Adatbázisok tipikusan on-line tranzakció feldolgozást végeznek (OLTP).
  - ◆ Rövid, egyszerű, gyakran feltett kérdések, mindegyik viszonylag kevés sort ad vissza válaszként.
  - ◆ Pl.: Web felületen keresztüli lekérdezések és válaszok, pénztárgépnél vásárlás, rep. jegy értékesítés

# OLAP

- ◆ Növekvő jelentőségű az OLAP jellegű lekérdezések.
  - ◆ Kisebb számú, de összetett , amelyek órákig is futhatnak.
  - ◆ A lekérdezések nem igénylik az abszolút időben pontos adatbázist.

# OLAP

- ◆ Általában az adatbázis nagyobb részét érintik az idetartozó tranzakciók.
  - ◆ Soros végrehajtás esetén leállhatnak az OLAP-műveletek → nem igazán tolerálható
  - ◆ Nem jó megengedni valamilyen típusú új adatok felvitelét, amikor éppen fut egy konkurens OLAP lekérdezés az ilyen típusú adatokon (ami pl. átlagolja ezeket)
- ◆ Korábban ezt nevezték adatbányászatnak.

# OLAP alkalmazások

- ◆ Általában az adattárházat a fogyasztásra vonatkozó adatokból állítják össze.
  - ◆ Akár terabájtnyi adatmennyiség arról, hogy melyik cikkből mennyi eladás történt
  - ◆ Előrejelzésre használható: fogyasztási adatok összesítése alapján valamilyen érdekes csoportok azonosítására

# OLAP Példák

1. Amazon elemzi vásárlói viselkedését azért, hogy olyan képernyő tartalmat jelenítsen meg, amely valószínűleg érdekli a vásárlót.
2. Wal-Mart-ot az érdekli, hogy melyik régióban melyik termék értékesítése növekszik



# Tipikus architektúra megoldások

- ◆ Az áruházláncok egyes áruházaik OLTP szinten dolgoznak.
- ◆ A helyi adatbázisokat éjszakánként feltöltik a központi adattárházba.
- ◆ Az adatelemzők az adattárházat OLAP elemzésekre használják fel.

# Csillag séma

- ◆ Csillag séma az adattárházak megszokott adatszerkezete. A következőkből áll:
  1. Tény tábla: olyan adatok kumulált tömege mint pl. az értékesítési adatok.
    - Általában csak „beillesztés”-re állított tábla.
  2. Dimenzió táblák: kisebb, általában statikus információkat tartalmaznak azokról az entitásokról, amelyekről tényeket tárolunk.

# Példa csillag sémára

- ◆ Tegyük fel, hogy az adattárházban fel akarunk jegyezni minden sör értékesítési adatot: a kocsmát, a sörfajtáját, az alkeszt, a napot, időpontot és fizetett árat.
- ◆ A ténytábla egy ilyen reláció lesz:  
Értékesítések(kocsmá, sör, alkesz, nap, idő, ár)

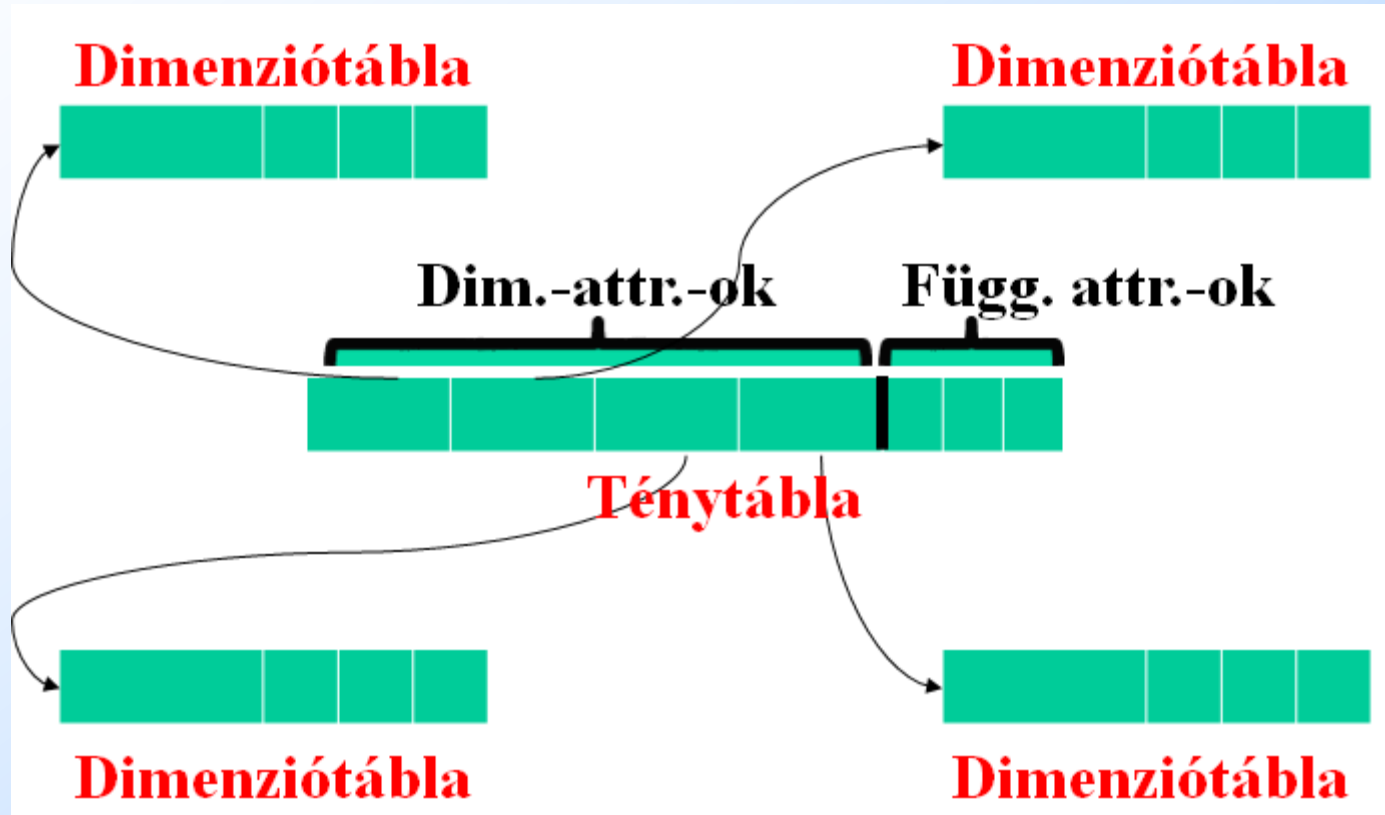
# Példa folytatás

- ◆ A dimenzió táblák a kocsmá, a sör és a alkesz adatait tartalmazzák

# Dimenziók és a függő attribútumok

- ◆ A ténytáblák attribútumainak két osztálya:
  1. Dimenzió attribútumok: A dimenzió táblák kulcsai. (Idegen kulcsok)
  2. A függő attribútum: A sorban a dimenzió attribútumok által meghatározott értékek

# Dimenziók és a függő attribútumok



# Példa függő attribútumok

- ◆ Az *ár* függő attribútum az *Értékesítés* relációban
- ◆ Amelyet a dimenzió attribútumok kombinációja határoz meg: a *kocsmá*, a *sör* és az *alkesz* valamint az időpont ( a *nap* és az *idő* attribútumok kombinációja)

# OLAP adatok többdimenziós nézete

- ◆ Gondolhatunk úgy is a ténytáblában tárolt adatokra, mintha többdim. térben vagy „kockában” lennének elrendezve
  - ◆ A kocka belsőpontjai ábrázolják az adatobjektumokat: pl. egy-egy gépkocsi eladás.
  - ◆ A dimenziók ennek az eladásnak a jellemzőit írják le.



# OLAP adatok többdimenziós nézete

- ◆ Az előbbi általában *nyers adatkockának* hívjuk, hogy megkülönböztessük az összetettebb *formális adatkockától* (ld. később bővebben)

# OLAP adatok többdimenziós nézete

- ◆ Ez utóbbi két dologban tér el:
  1. A dimenziók részhalmazaira vett összesítéseket is tartalmazza az adatokon túl.
  2. Az itteni belső pontok az adatobjektumoknak egy kezdeti összesítését is ábrázolhatják (pl. nem az összes sörnek az eladásait ábrázoljuk, hanem gyártónként összesítve)

# Adattárház készítési módok

- 1. ROLAP* = relációs OLAP. Relációs adatbáziskezelő rendszer olyan hangolása, amely a csillag sémát támogatja.
- 2. MOLAP* = többdimenziós OLAP: specializált adatbáziskezelő használata, amely pl. az adatkocka adatszerkezetet támogatja.

# ROLAP technikák

1. *Bitmap indexek:* a dimenzió táblák mindegyik kulcsértékére (pl. minden gyártóra a *Sör* relációban) egy bit vektor létrehozása, amely megmondja, hogy mely sorok tartalmazzák ezt az értéket.

Sör	Gyártó
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
Miller Lite	Miller Brewing Co.
Miller Genuine Draft	Miller Brewing Co.
Miller High Life	Miller Brewing Co.

Gyártó='A.-B.'	Gyártó='M.'
1	0
1	0
1	0
0	1
0	1
0	1

# ROLAP technikák

- 2. Materializált nézetek:* az olyan nézeteket, amelyek több lekérdezés megválaszolásához hasznosak magában az adattárházban eltárolják.

# Tipikus OLAP lekérdezések

- ◆ Az OLAP lekérdezések gyakran egy csillag összekapcsolással kezdődnek: a ténytábla és a dimenzió táblák természetes összekapcsolásával.

```
SELECT *  
FROM Értékesítések, Kocsmák, Sörök,  
Alkeszek  
WHERE Értékesítések.kocsmasma =  
Kocsmák.kocsmasma AND Értékesítések.sör  
= Sörök.sör AND Értékesítések.alkesz  
= Alkeszek.alkesz;
```

# Tipikus OLAP lekérdezések 2

## ◆ Tipikus OLAP lekérdezések:

1. Egy csillag séma összekapcsolással kezdődik.
2. Leválogatják a fontos sorokat, a dimenzió táblák adatai alapján
3. Egy vagy több dimenzió alapján csoportosítjuk.
4. Az eredmény egyes attribútumait összegezzük

# Példa: OLAP lekérdezés

- ◆ Palo Alto mindegyik kocsmájára keressük meg az Anheuser-Busch által gyártott mindegyik sörre az összes eladás értékét
- 1. Szűrő: *cím* = "Palo Alto" és *gyártó* = "Anheuser-Busch".
- 2. Csoportosítás: kocsmák és sör.
- 3. Összesítés: az *ár* összege



# Példa: SQL-ben

```
SELECT kocσμα, sör, SUM(ár)
FROM Értékesítések NATURAL JOIN
     Kocsmák NATURAL JOIN Sörök
WHERE cím = 'Palo Alto' AND
     gyártó = 'Anheuser-Busch'
GROUP BY kocσμα, sör;
```

# Materializált nézetek használata

- ◆ Az előbbi példa lekérdezés közvetlen végrehajtása az *Értékesítések* és a dimenzió táblák segítségével túl hosszú ideig tartana.
- ◆ Ha egy materializált nézetet hozunk létre, amely elegendő információt tartalmaz sokkal gyorsabban lehetne megválaszolni

# Példa: Materializált nézetek használata

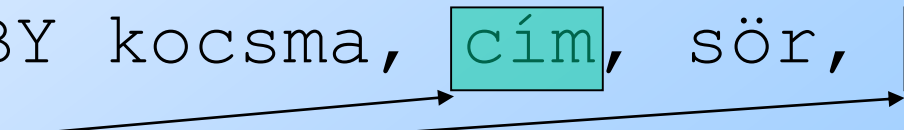
- ◆ Mely nézetek segítenék a lekérdezést:
- ◆ Kulcs kérdések:
  1. Össze kell kapcsolni minimum az *Értékesítések*, *Kocsmák*, és *Sörök* táblákat.
  2. Csoportosítani kell legalább *kocsmá* és *sör* attr. szerint.
  3. Sem a Palo-Alto-i kocsmákat (*sör*) sem Anheuser-Busch söreit (*sör*) nem szabad kihagyni.
  4. Nem szabad lehagyni az eredményből, vetítéssel, sem a *cím* sem a *gyártó* attribútumokat.

# Példa --- folytatás

Itt a materializált nézet:

A funkcionális függések miatt nincs igazi csoportosítás

```
CREATE MATERIALIZED VIEW KCSGE (kocsma,  
cím, sör, gyártó, értékesítés) AS  
SELECT kocsma, cím, sör, gyártó,  
SUM(ár) értékesítés  
FROM Értékesítések NATURAL JOIN  
Kocsmák NATURAL JOIN Sörök  
GROUP BY kocsma, cím, sör, gyártó;
```



Mivel kocsma -> cím és sör -> gyártó, ezért nincs igazi csoportosítás  
Szükségünk van címre és gyártóra a SELECT-ben.

# Példa --- Lezárás

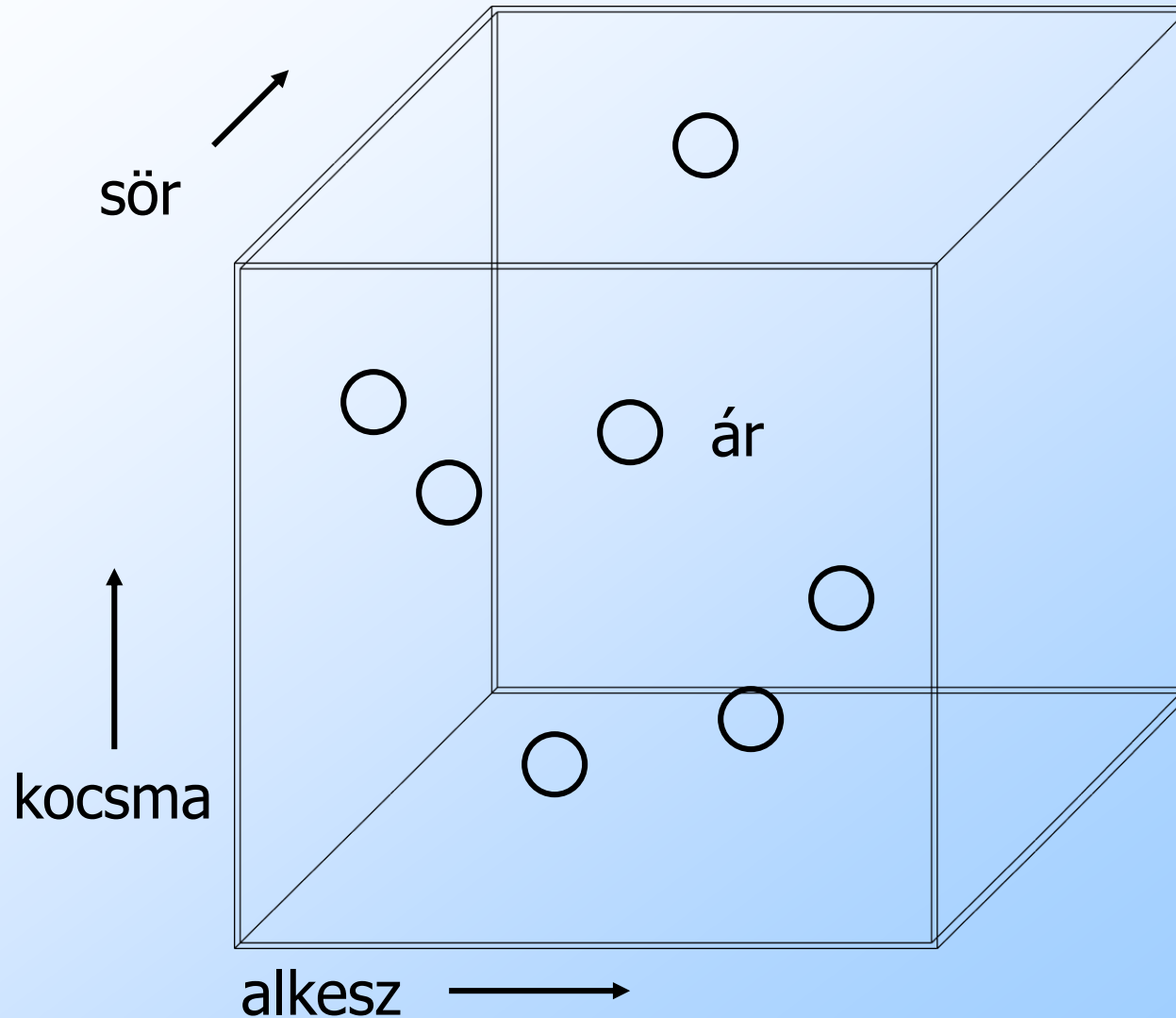
BABMS materializált nézet lekérdezése:

```
SELECT koccsma, sör, értékesítés  
FROM KCSGE  
WHERE cím = 'Palo Alto' AND  
       gyártó = 'Anheuser-Busch';
```

# MOLAP és adatkockák

- ◆ A dimenzió táblák kulcsai a hiper-kocka dimenziói:
  - ◆ Példa: *Értékesítések* adataira, négy dimenzió: *kocsmák, sörök, alkeszek, és idő.*
- ◆ A függő attribútumok (pl. *ár*) a kocka „belső” pontjaiban jelennek meg

# Vizualizáció -- Adatkockák

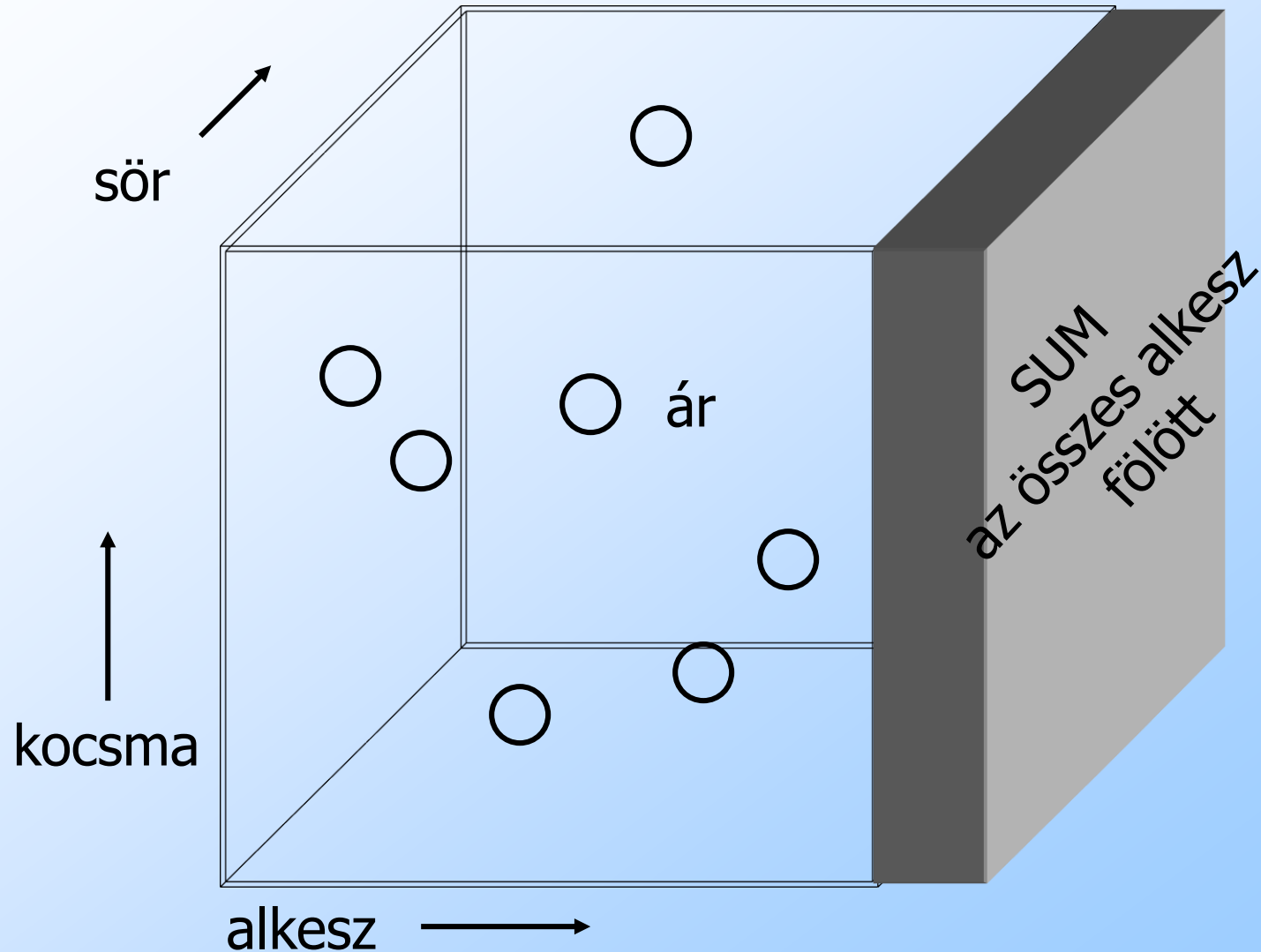


# Kocka oldalai, szélei

- ◆ Az adatkocka tartalmazhat összesítéseket (tipikusan SUM) a kocka oldalai szerint
  - ◆ A már említett példa: a sörök dimenzióját kicseréljük a gyártóra
- ◆ A kocka szélei tartalmazhatnak összesítést egy dimenzióban, két dimenzióban, stb.



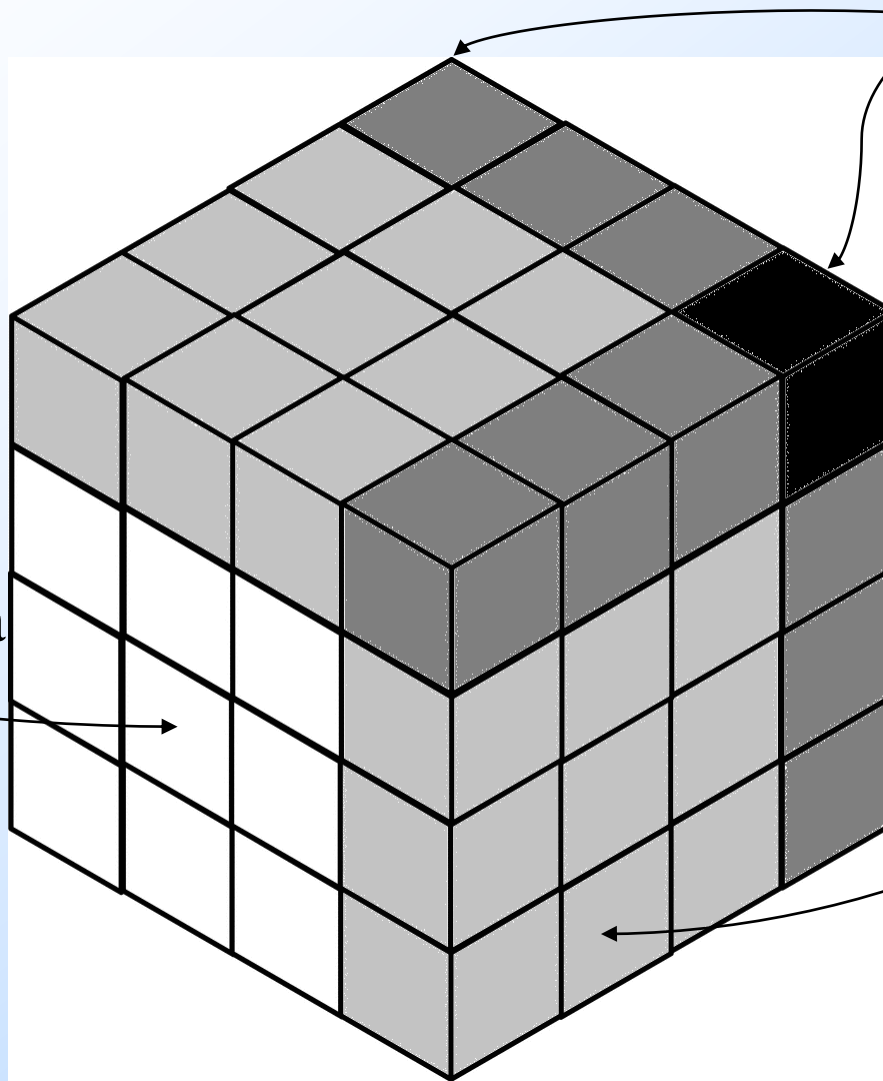
# Vizualizáció --- Adatkocka összesítéssel



# Példa

- ◆ A példánk 4 –dimenziós adatkockája tartalmazza az *árak* összegét, minden kocsmára, sörre, alkeszre és idő egységre (pl. napra)
- ◆ De tartalmazhatná az értékesítési adatokat, az *árak* összegét minden kocsmá-sör párosra, minden kocsmá-alkesz-nap hármásra, stb.

Eredeti adatkocka



A kocka határainak bővítése a dim.-ok minden lehetséges kombinációján kiszámított összesítéseknek megfelelő módon

# Az adatkocka szerkezete

- ◆ Képzeljük azt, hogy mindegyik dimenzió tartalmaz egy további értéket (\*) [*A szokásos vélelmezett „tetszőleges” érték értelmezéssel*].
- ◆ Egy olyan pont, amelynek koordinátái között egy vagy több „\*” szerepel azt jelenti, hogy azok fölött a dimenziók fölött, amelyeknek az értéke ebben a sorban (adatkocka pontban) „\*” összegzést hajt végre.
- ◆ Pl. Értékesítések(“Joe’s Bar”, “Bud”, \*, \*, *értékesítés*), minden alkeszre vonatkozóan, az összes eltöltött időt figyelembe véve, amikor Bud-t fogyasztottak Joe kocsmájában (“Joe’s Bar” ) a fogyasztás teljes összegét (*értékesítés*) adja meg.

# Lefűrás

- ◆ Lefűrás= „finomabb összegzés” = az összegzéseket finomabb alkotórészeire bontjuk.
- ◆ Példa: Ha kiderült, hogy Joe kocsmája nagyon kevés Anheuser-Busch sört adott el.
- ◆ Bontsuk fel az értékesítési adatokat az egyes Anheuser-Busch sörfajták szerint.

# Felgörgetés (felösszegzés)

- ◆ Felgörgetés = összegzés egy vagy több dimenzió mentén
- ◆ Példa: legyen egy olyan táblánk, amelyben minden alkeszről azt tároljuk, hogy mennyi *Bud sört* fogyasztott el az egyes kocsmákban, görgessük fel ezt egy olyan táblába, amelyik megadja minden alkeszre azt, hogy mennyi *Bud sört* fogyasztott el

# Materializált adatkocka nézetek

- ◆ Adatkockák létrehozására olyan materializált nézeteket lehet használni, amelyek egy vagy több dimenzióban összegzéseket tartalmaznak.
- ◆ Az egyes dimenziókat nem kell teljes mértékben összegezni – az egyik lehetőség, hogy a dimenzió tábla bizonyos attribútumai szerint összegzünk.

# Példa

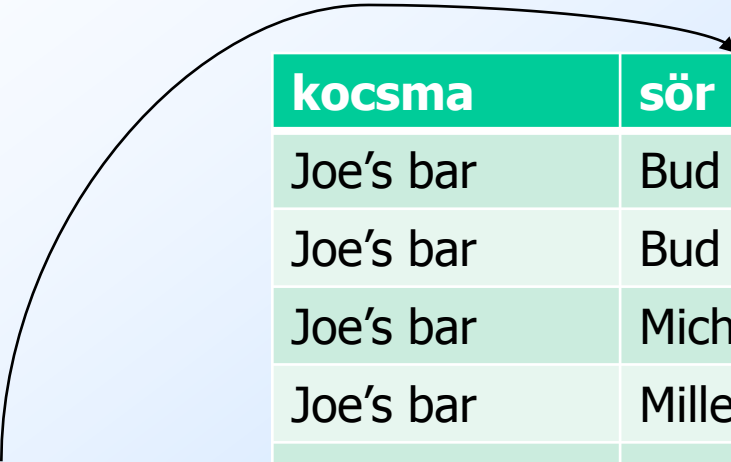
- ◆ Az értékesítés materializált nézete (*Értékesítések*), adatkocka a következő lehet:
- ◆ Alkeszek alapján teljes összegzés.
- ◆ A sör fajták alapján nincs összegzés
- ◆ Idő tekintetében hetek szerinti összegzés
- ◆ A kocsmák városa szerinti összegzés.



# Kockaművelet SQL-ben

- ◆ Az SQL támogatja a kockaműveletet: `WITH CUBE` (illetve PostgreSQL-ben, Oracle-ben: `GROUP BY CUBE (A, B)`)
- ◆ Így nem csak az egyes csoportokra vonatkozó sorokat kapjuk vissza, hanem azokat is, amelyek összesítéseket fejeznek ki a csoportosításban résztvevő egy/több dimenzióra nézve
- ◆ A soroknál a \*-ot a NULL jelzi

# Példa



kocsmá	sör	ár
Joe's bar	Bud	2.5
Joe's bar	Bud Lite	3.5
Joe's bar	Michelob	3.0
Joe's bar	Miller Lite	2.75
Joe's bar	Miller Genuine Draft	4.5
Joe's bar	Miller High Life	3.25
Sue's bar	Bud	2.0
Sue's bar	Bud Lite	2.25
Sue's bar	Miller High Life	3.75
Sue's bar	Miller Lite	2.75
Sue's bar	Dreher	4.0

Materializált nézet:  
Értékesítések

```
SELECT kocsmá, sör, SUM(ár)
FROM Értékesítések
GROUP BY kocsmá, sör WITH CUBE
```

kocsmá	sör	SUM
Joe's bar	Bud	2.5
Joe's bar	Bud Lite	3.5
Joe's bar	Michelob	3.0
Joe's bar	Miller Lite	2.75
Joe's bar	Miller Genuine Draft	4.5
Joe's bar	Miller High Life	3.25
Sue's bar	Bud	2.0
Sue's bar	Bud Lite	2.25
Sue's bar	Miller High Life	3.75
Sue's bar	Miller Lite	2.75
Sue's bar	Dreher	4.0
Joe's bar	NULL	14.75
Sue's bar	NULL	19.5
NULL	Bud	4.5
NULL	Bud Lite	5.75
NULL	Michelob	3.0
NULL	Miller Lite	5.5
NULL	Miller Genuine Draft	4.5
NULL	Miller High Life	7.0
NULL	Dreher	4.0
NULL	NULL	19.5

# Adatbányászat

- ◆ Az adatbányászat az olyan adatfeldolgozásokat jelenti, amelyek nagy adathalmazokat összegeznek valamilyen szempontból hasznos módon.
- ◆ Példák:
  1. Az összes Web lap klaszterezése témák szerint.
  2. A bankkártyák csalásra történő használatának jellemzőinek feltárása

# Bevásárló kosár

- ◆ A relációs adatforrásokból az egyik tipikus adatbányászati feladat a *bevásárló kosár* = olyan bevásárlási tételek listája, amiket egy fogyasztó megvásárolt.
- ◆ A bevásárlási adatok egyik összegzése a *gyakran előforduló tételhalmazok* = olyan áru tételek, amelyek gyakran fordulnak elő együtt.

# Példa

- ◆ Ha valaki gyakran vásárol hamburgert és ketchup-ot együtt és szalma krumplit vesz még hozzá.
- ◆ Akkor csináljunk egy árleszállítást a hamburgerre, viszont emeljük meg a ketchup árát.

# Gyakori párosítások feltárása

- ◆ Az egyszerű eset, amikor csak a gyakori párosításokat akarjuk megtalálni.
- ◆ Tegyük fel, hogy az adatok a *Baskets(basket, item)* relációban vannak.
- ◆ Az *s támogató küszöbérték* az olyan bevásárló kosarak minimális száma, amelyben egyrészt a minket érdeklő termék párosítások jelennek meg, és másrészt ha ezt a küszöb értéket túllépi a bevásárló kosarak száma, akkor érdekesek lesznek számunkra.

# Gyakori párosítások SQL-ben

Keressük azokat a bevásárló kosár párosokat, amelyek ugyanarra a kosárra vonatkoznak, de az árutételek különböznek

Az első tételnek meg kell előznie a másikat azért, hogy ne számoljuk kétszer ugyanazt a párt.

```
SELECT b1.item, b2.item  
FROM Baskets b1, Baskets b2  
WHERE b1.basket = b2.basket  
AND b1.item < b2.item  
GROUP BY b1.item, b2.item  
HAVING COUNT(*) >= s;
```

Dobjuk el azokat a termékeket, amelyek nem jelennek meg legalább  $s$ -szer

Hozzunk létre egy csoportot minden olyan pár termékre, amelyek legalább az egyik kosárban megjelenik



# A-Priori Trükk --- 1

- ◆ Az egyszerű megvalósítást a *Baskets* reláció önmagával történő összekapcsolásával lehetne megoldani
- ◆ Az *a-priori algoritmus* felgyorsítja a lekérdezést, a következőképpen, egy  $\{i,j\}$  csak akkor éri el a támogató küszöbértéket,  $s-t$ , ha mind  $\{i\}$  mind  $\{j\}$  vagyis mindkettő eléri.

# A-Priori Trükk --- 2

- ◆ Használjunk egy materializált nézetet a gyakori termékek nyilvántartására.

```
INSERT INTO Baskets1 (basket, item)
```

```
SELECT * FROM Baskets
```

```
WHERE item IN (
```

```
    SELECT ITEM FROM Baskets
```

```
    GROUP BY item
```

```
    HAVING COUNT(*) >= s
```

```
);
```

Termékek, amelyek  
legalább  $s$  kosárban  
megjelennek

# A-Priori Algoritmus

1. Készítsünk egy *Baskets1* materializált nézetet.
2. A nyilvánvaló lekérdezést futtassuk a *Baskets1-en*, *Baskets* helyett.
  - *Baskets1* lekérdezése olcsó, mivel nincs benne összekapcsolás
  - *Baskets1* –nek kevesebb sora van (val.szeg) mint *Baskets*-nek
  - A futási idő az összekapcsolásban érintett sorok számának négyzetével arányosan csökken.

# Példa

## ◆ Tegyük fel:

1. Egy supermarket 10,000 terméket árusít.
2. Az átlagos kosár tartalma 10 tétel.
3. A támogatási küszöbérték kosarak számának 1%-a

## ◆ Ezért legfeljebb a tételek $1/10$ tekinthető gyakorinak.

## ◆ Valószínűleg, egy kosárban a tételek kisebb része tekinthető gyakorinak → 4-szeres gyorsuláshoz vezet ez a feltételezés.