Modelowanie konceptualne i fizyczne

Budowanie bazy danych dla “rzeczywistego” problemu

Zagadnienie rzeczywiste

Zagadnienie rzeczywiste (cd.)

- masz magazyn na to, co kupiłeś/zebrałeś/wyprodukowałeś
- zebrane plony/produkty możesz sprzedawać
- masz zasoby waluty (żetony, gotówkę)
- w grze zdobywasz kolejne poziomy i tytuły
- waluta jest potrzebna, by kupować narzędzia, nasiona, urządzenia
- w grze masz sąsiadów - znajomych
- można przesyłać sobie prezenty
- można przesyłać sobie wiadomości

Struktura zagadnienia

- Obiekty: na farmie są: pola, drzewa, kwiaty, zwierzęta, urządzenia, dekoracje.
- Procesy: obiekty mogą rosnąć (zmieniać wygląd), poruszać się, produkować coś (owoce, plony, mąkę, jaja lub mleko):
  - wzrost kwiatu/drzewa składa się z kilku (niewielu) etapów; obiekt rosnący zmienia wygląd; wzrost jest samorzutny;
  - produkcja zwierząt jest samorzutna, ale wymaga odpowiednich obiektów (obory, owczarni, kurnika); zwierzeta mogą się poruszać;
  - produkcja w przedsiębiorstwach wymaga składników (zboże, kawa, mąka) i jest uruchamiana przez użytkownika; oprócz składników wymaga waluty;
  - uprawa składa się z: orania, siania, zbierania plonów.
- Katalizatory: możesz mieć zasoby (narzędzia, nawozy) przyśpieszające procesy;
- Zasoby: w grze zdobywasz:
  - walutę, poziomy i tytuły, katalizatory
  - plony i produkty
  - obiekty (zakupione i umieszczone na farmie lub w magazynie)
- Związki: gracze mogą być sąsiadami

Rozpoznanie struktury zagadnienia

- Wyróżniamy *encje* (obiekty), tj: użytkownik, farma, pole, drzewo, urządzenia, zwierzęta, narzędzia, produkty;
- Dla każdego zbioru encji określamy *atrybuty*, które będą przechowywane w bazie, tj. nazwa użytkownika, obraz i pozycja obiektu, etap rozwoju;
- Znajdujemy *związki* pomiędzy encjami tj.: produkt jest produkowany przez rządzenie, obiekt znajduje się na farmie, produkt jest potrzebny do wytworzenia innego produktu, gracze są sąsiadami.

Po wyróżnieniu encji, atrybutów i związków możemy narysować **diagram E-R**.

Diagram E-R (zbiory encji i atrybuty)

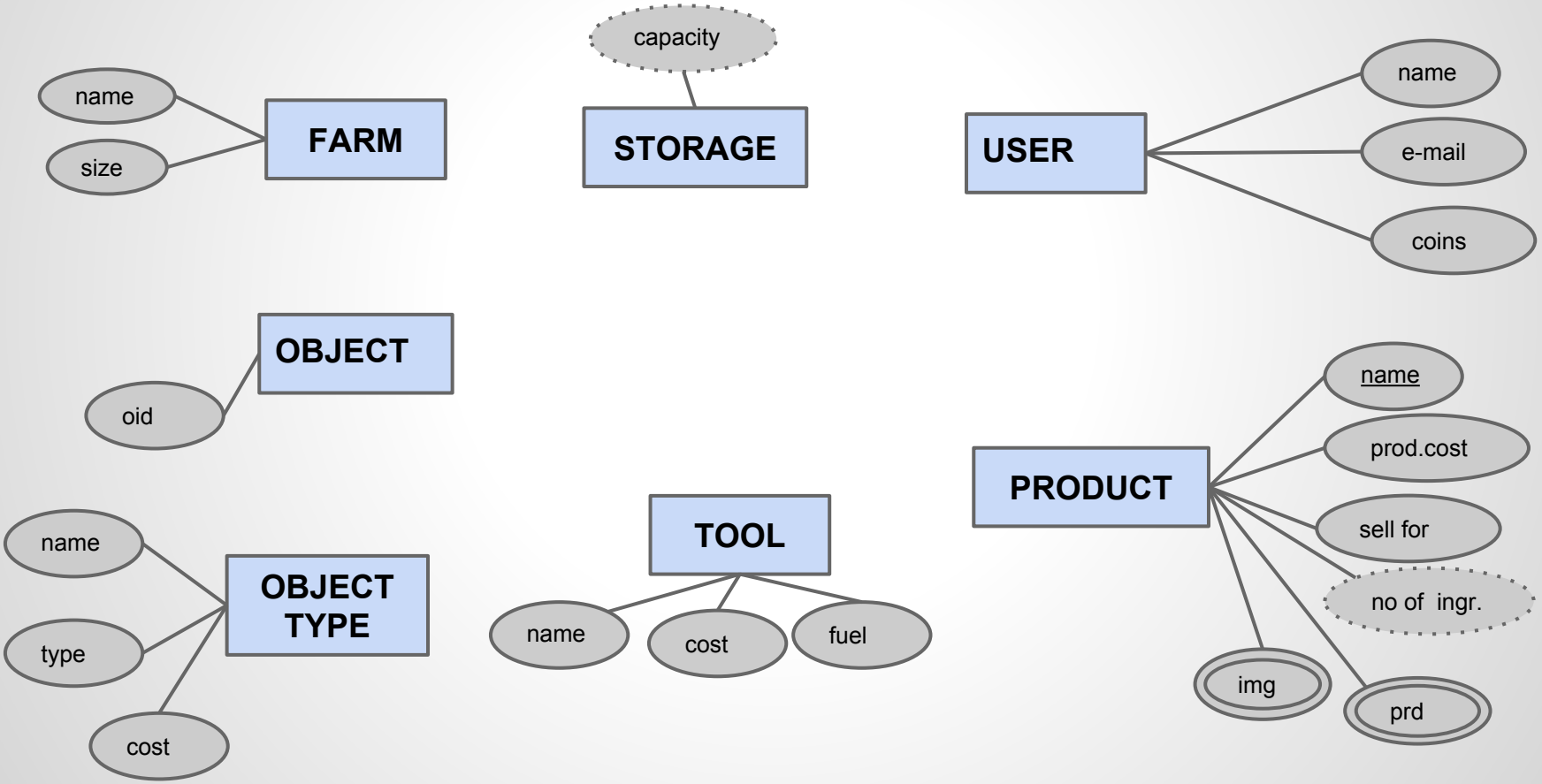


Diagram E-R (zbiory encji i atrybuty)

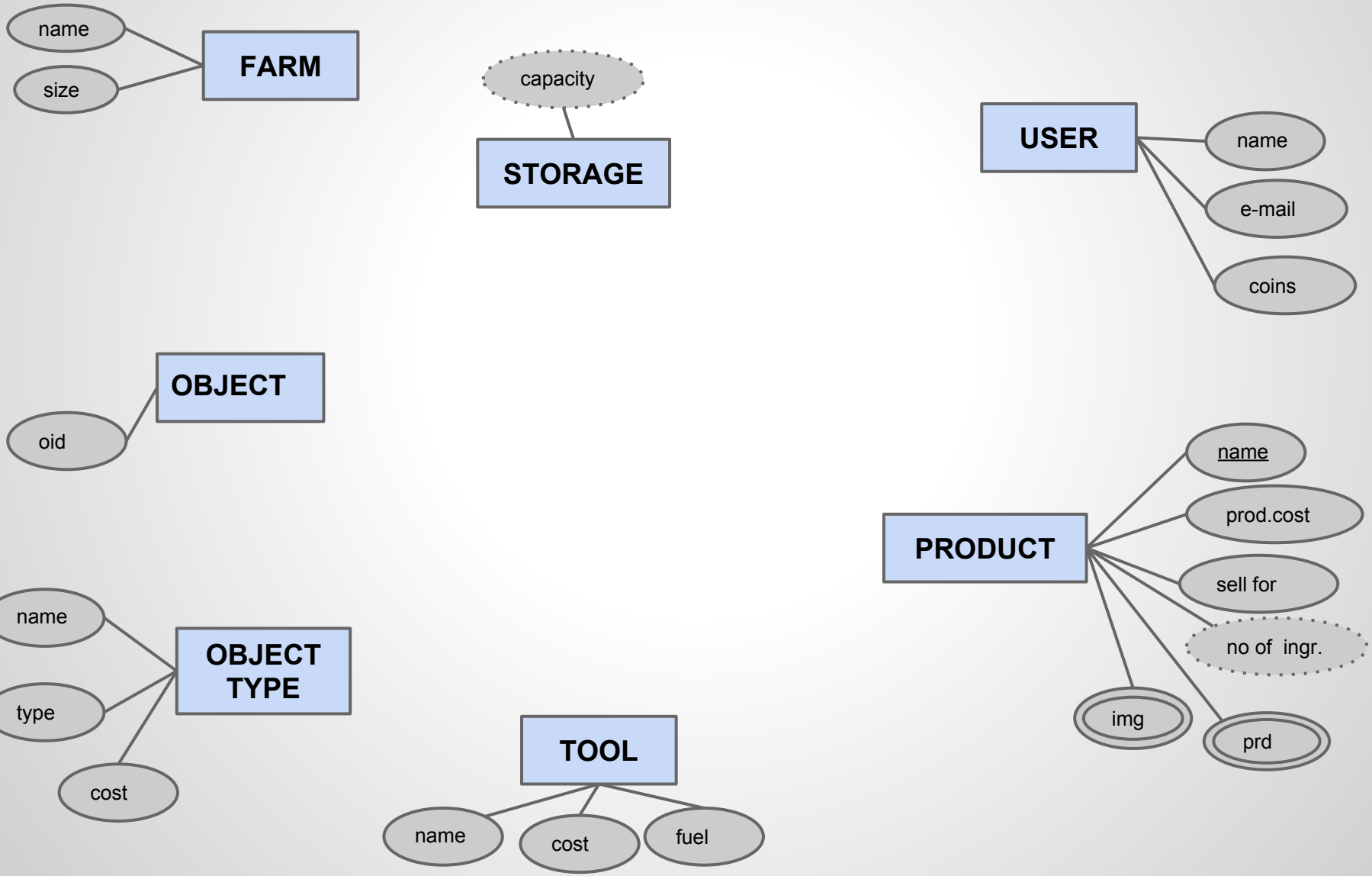


Diagram E-R (związki)

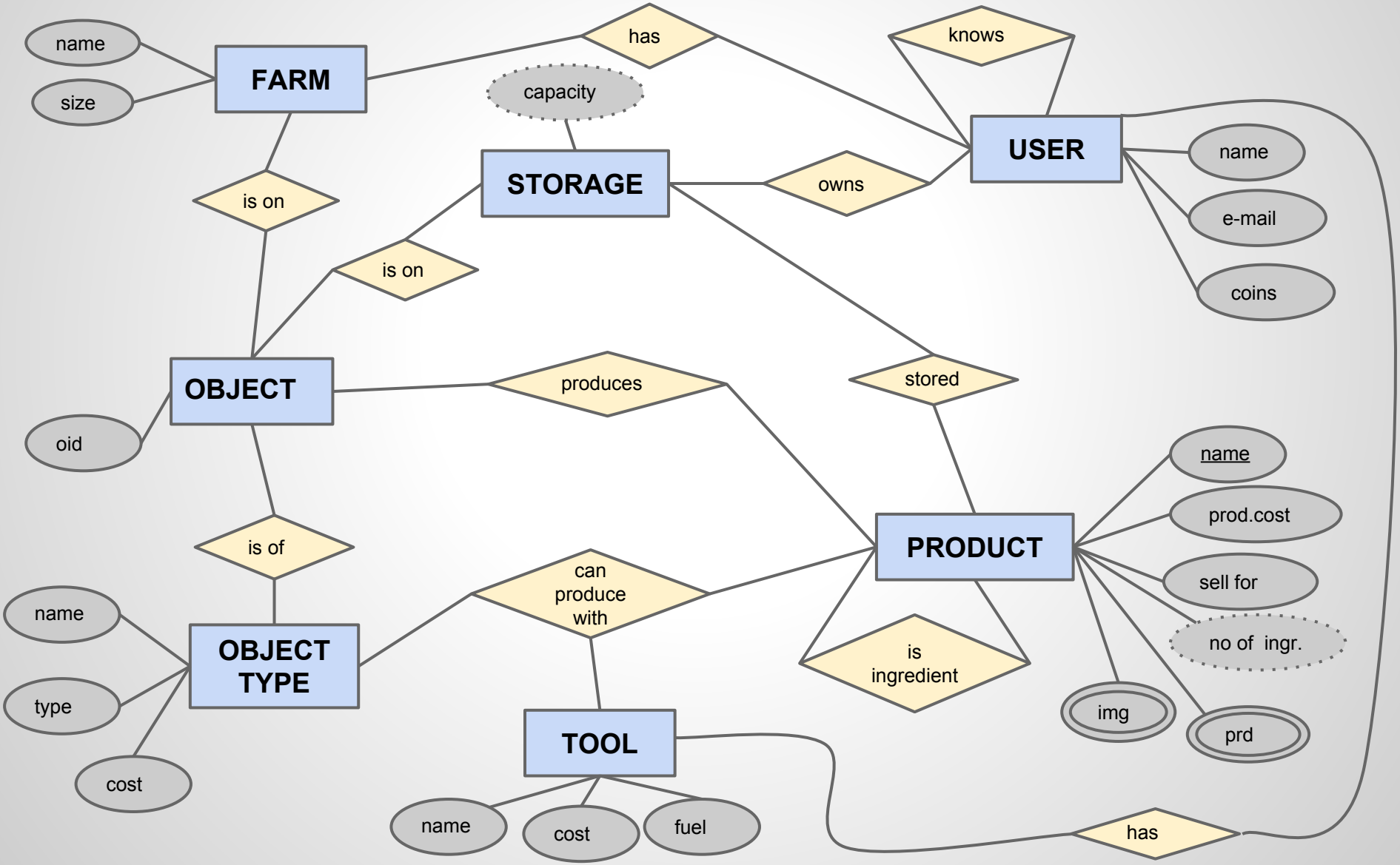


Diagram E-R (atrybuty związków)

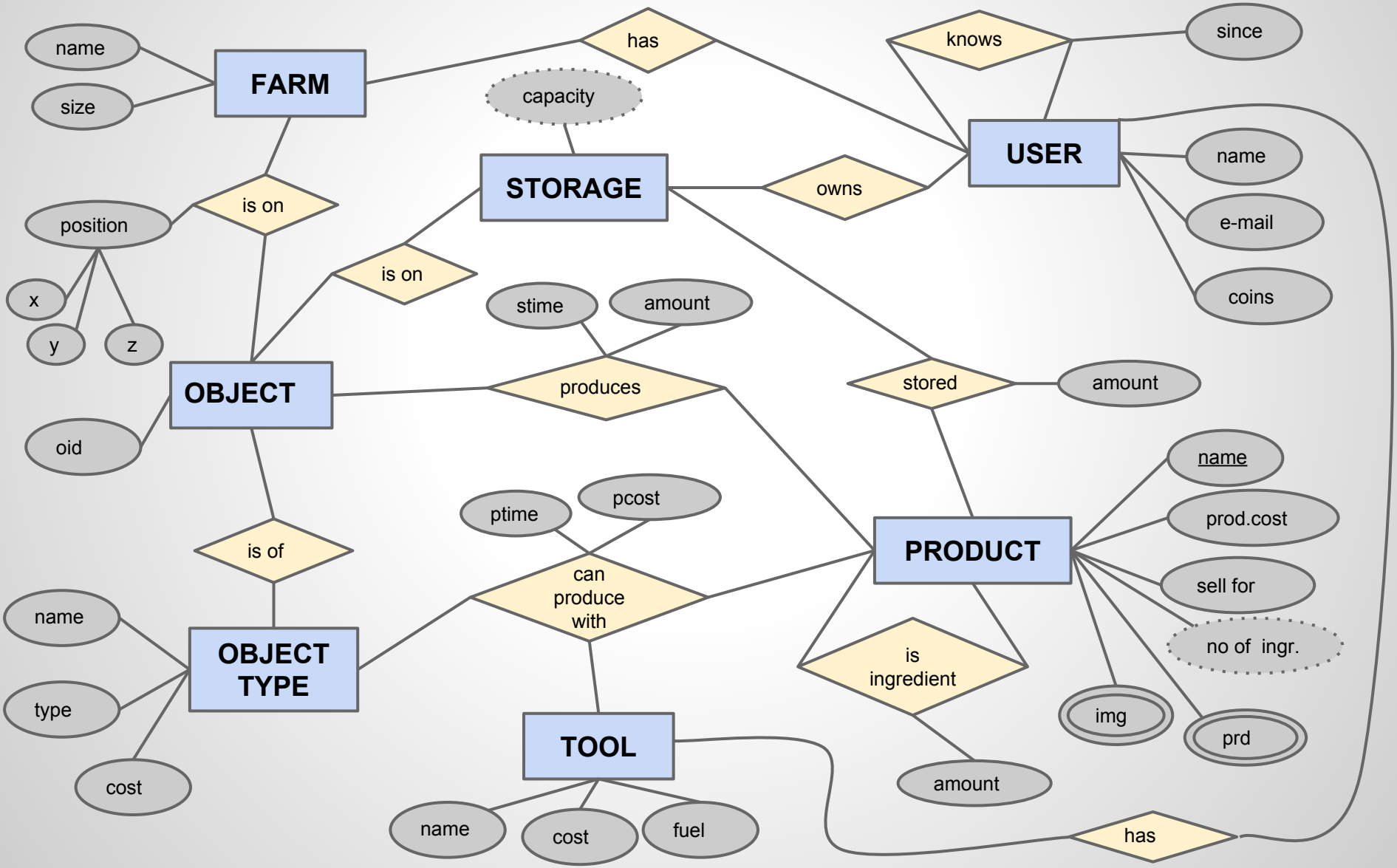


Diagram E-R (rodzaje związków)

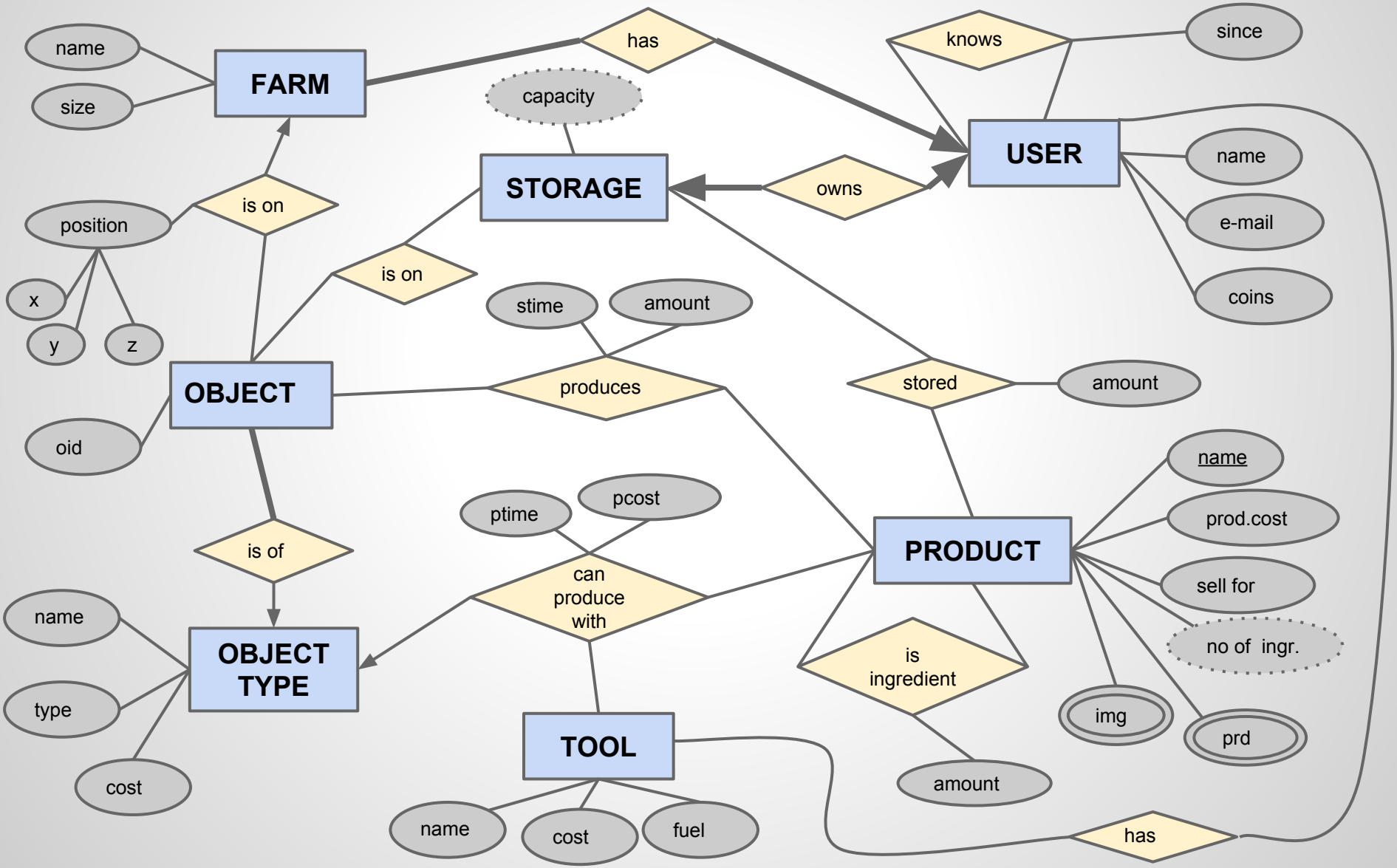
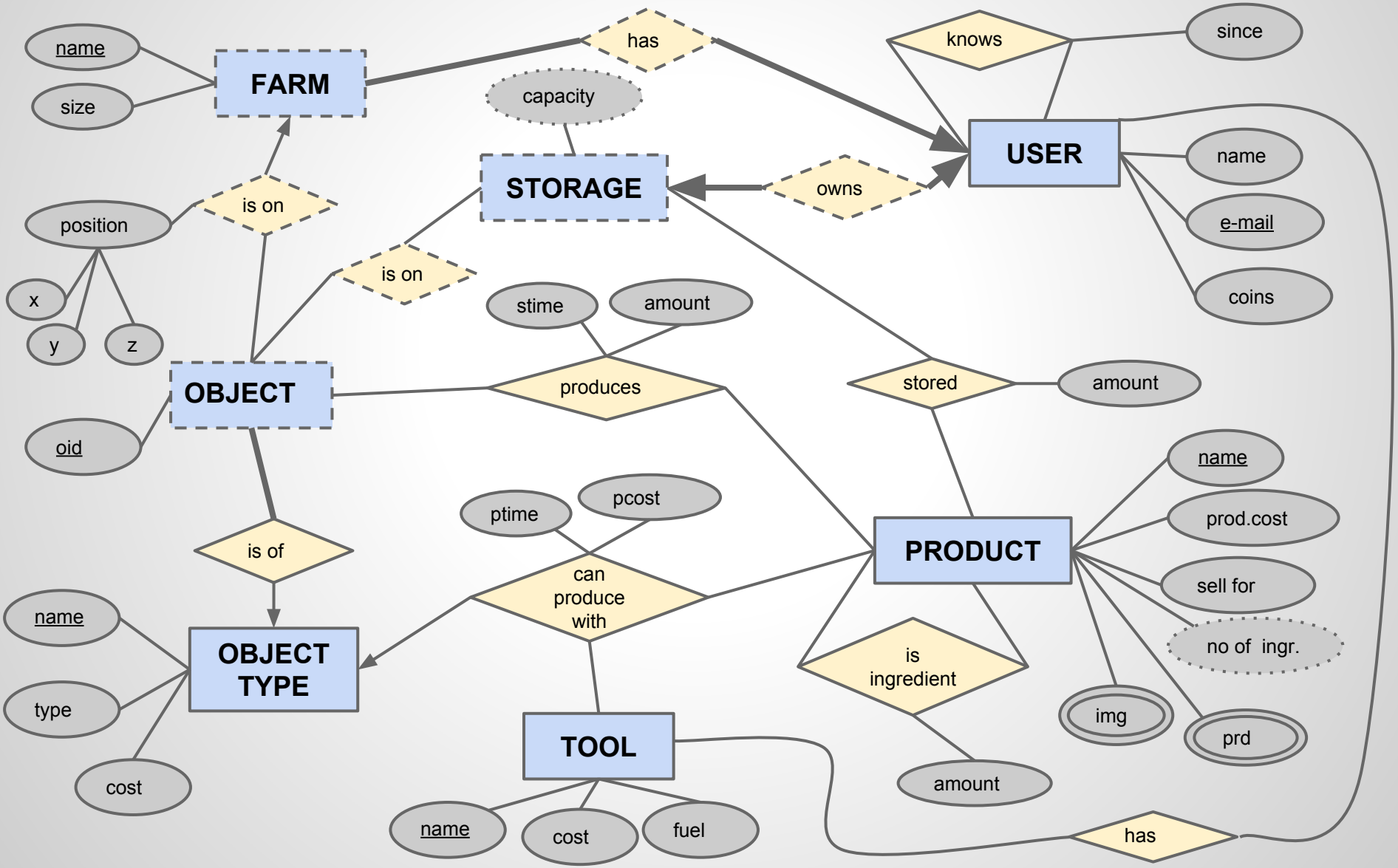
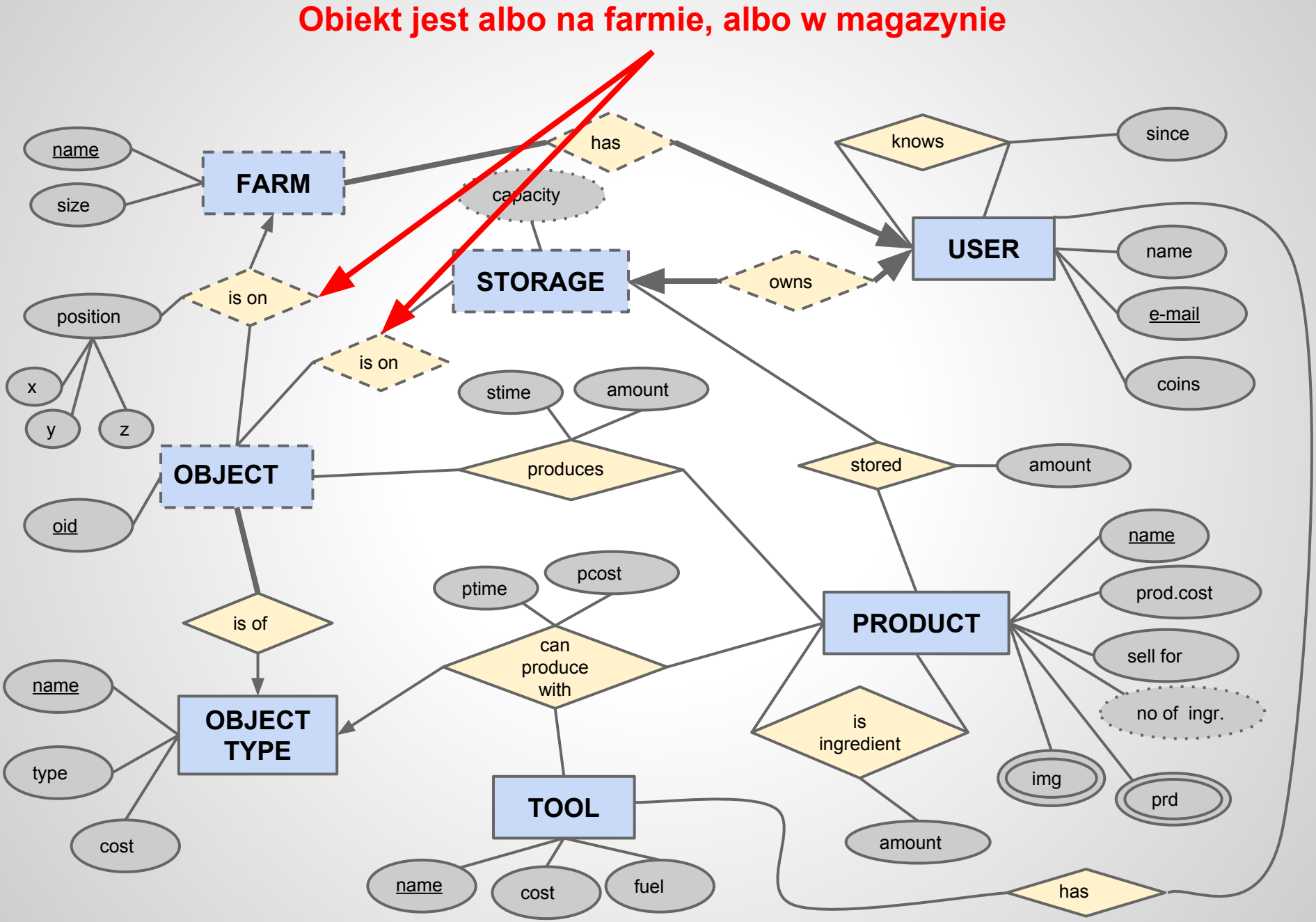
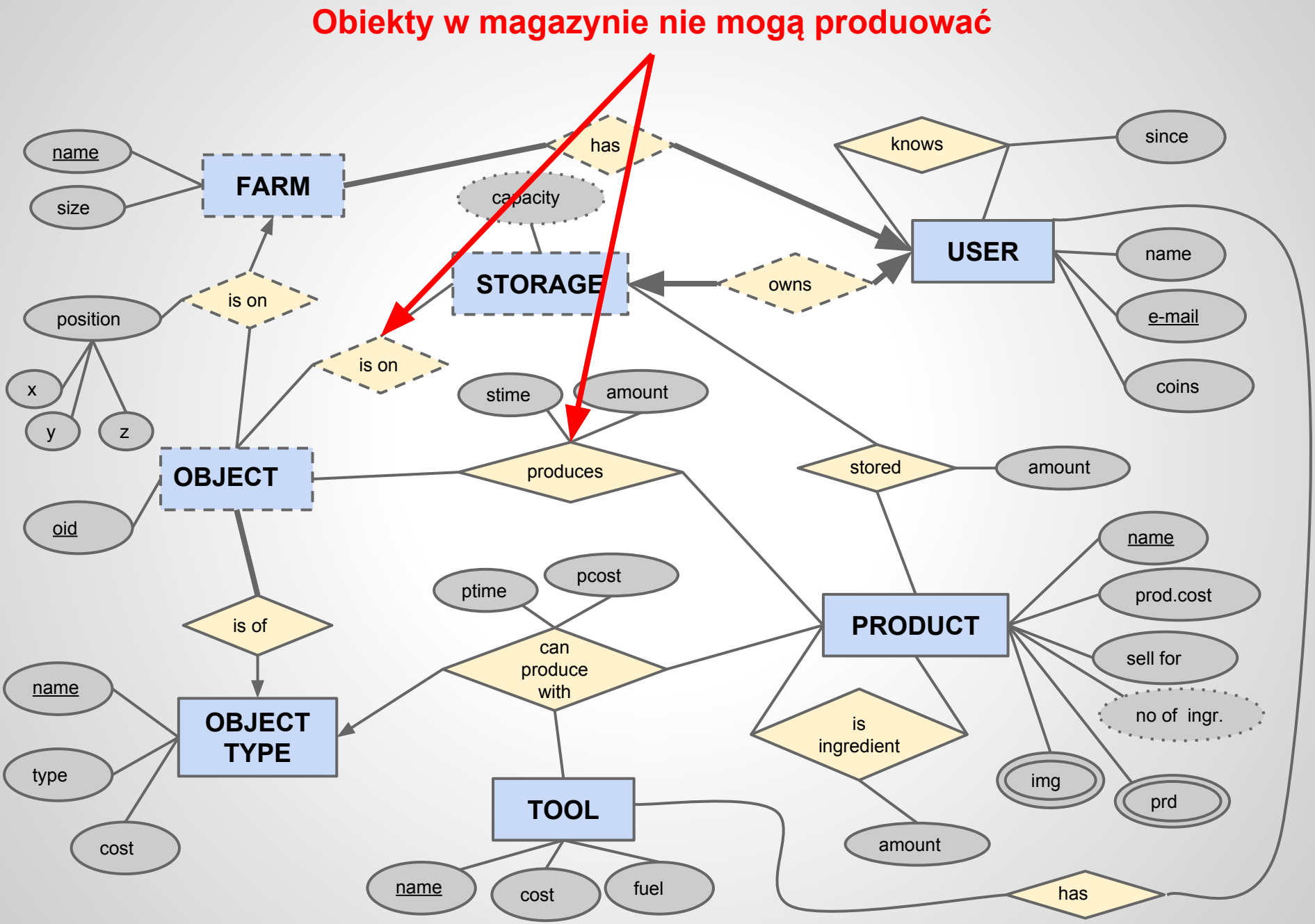


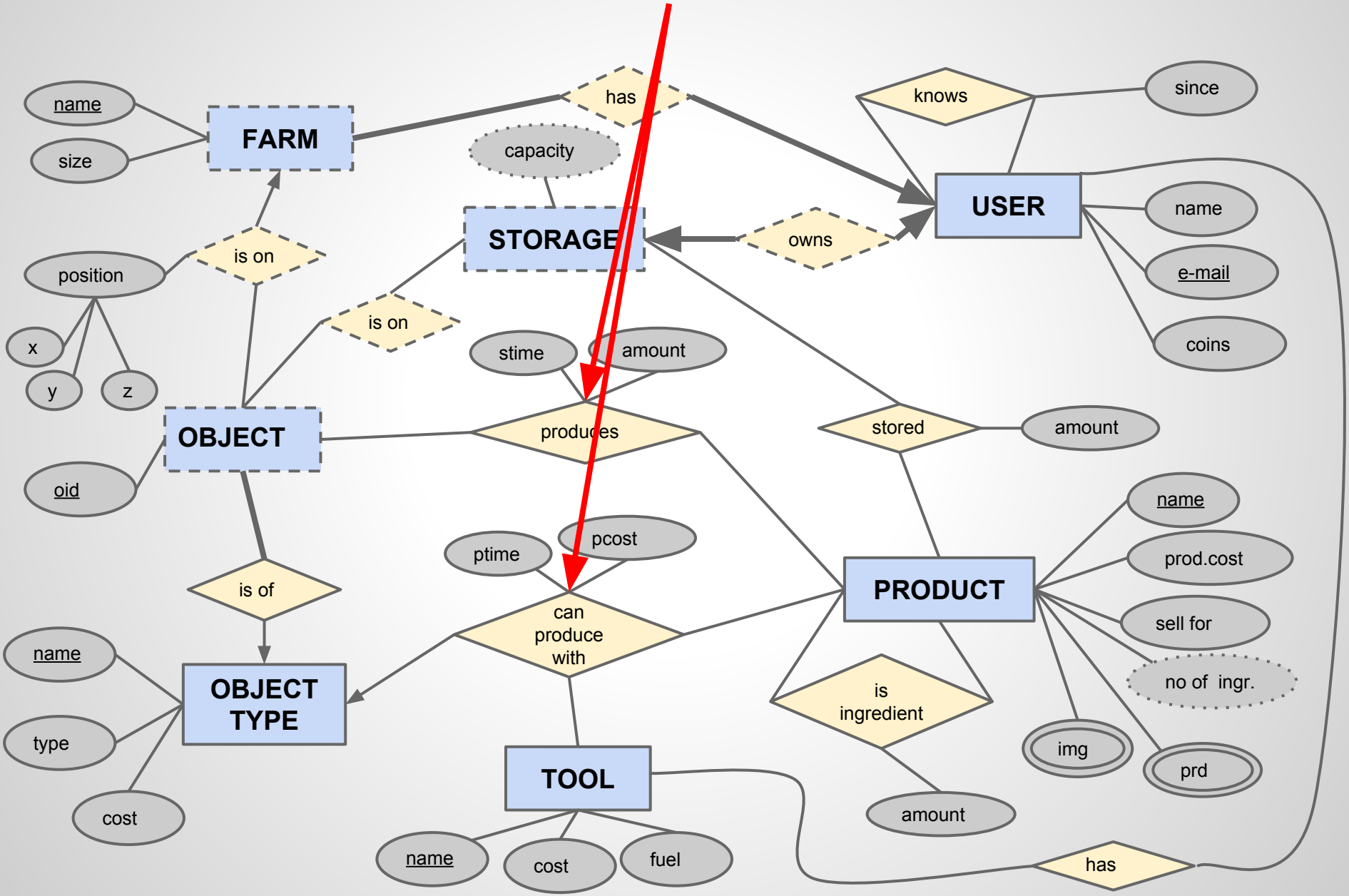
Diagram E-R (klucze i słabe zbiory encji)

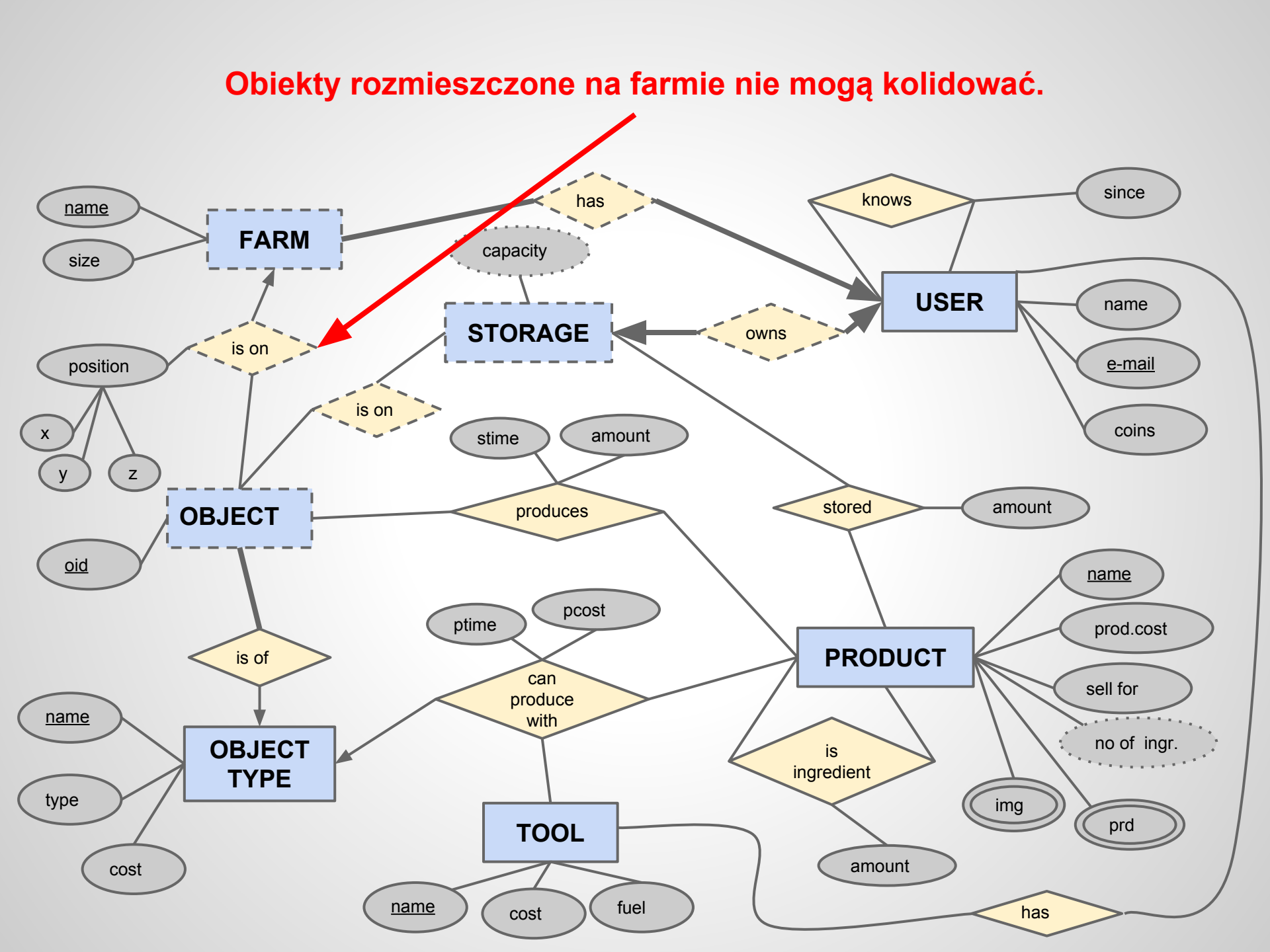



Obiekt jest albo na farmie, albo w magazynie
name
size
FARM
has
capacity
knows
since
USER
name
e-mail
coins
STORAGE
owns
is on
position
x
y
z
is on
stime
amount
OBJECT
produces
stored
amount
oid
is of
ptime
pcost
can produce with
PRODUCT
name
prod.cost
sell for
no of ingr.
name
type
cost
OBJECT TYPE
is ingredient
img
prd
amount
TOOL
name
cost
fuel
has

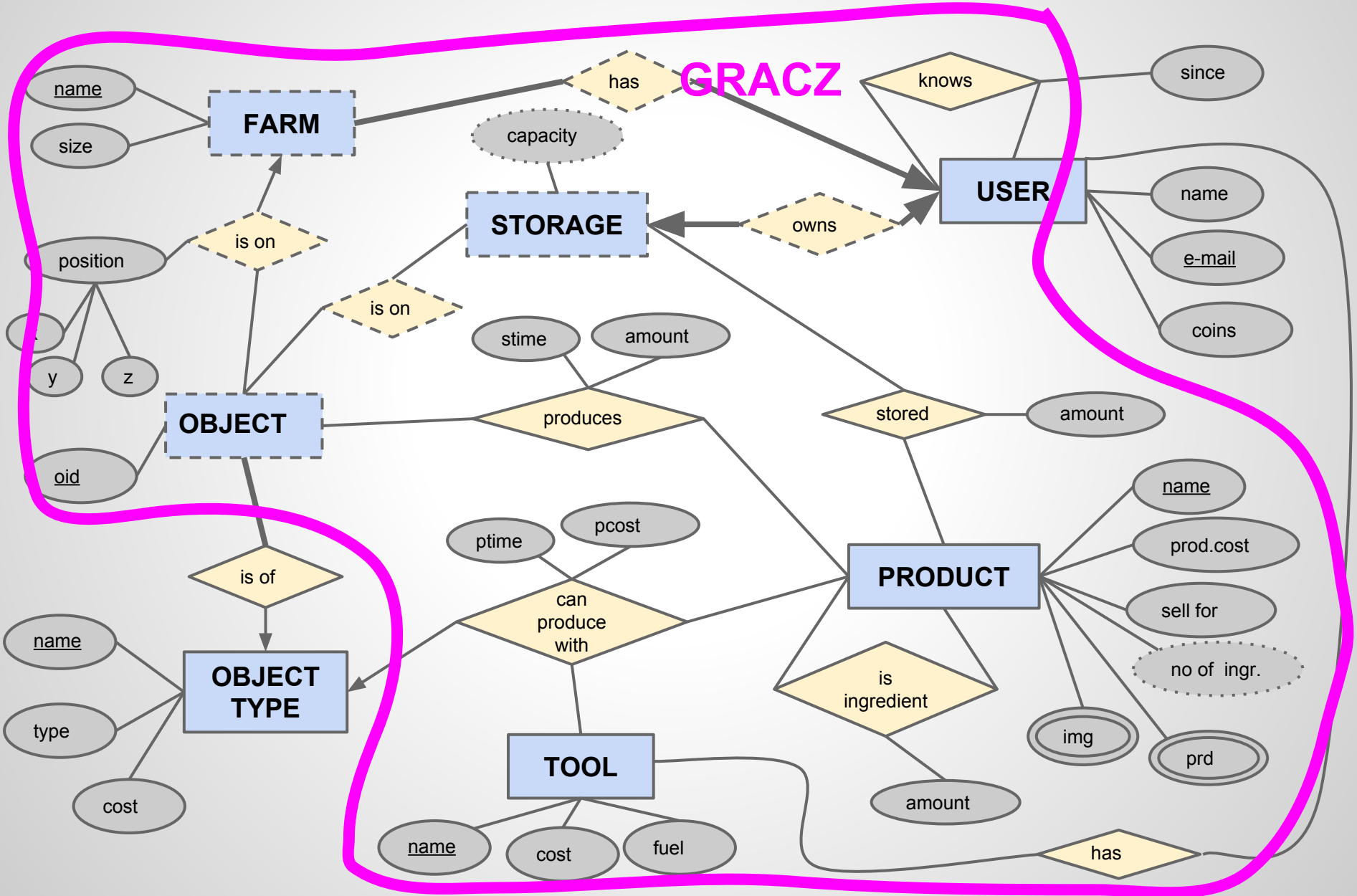

Obiekty w magazynie nie mogą produować
name
size
FARM
has
capacity
knows
since
USER
name
e-mail
coins
STORAGE
owns
is on
position
x
y
z
is on
stime
amount
produces
OBJECT
oid
stored
amount
name
prod.cost
sell for
no of ingr.
img
prd
PRODUCT
ptime
pcost
is of
can
produce
with
name
type
cost
OBJECT
TYPE
is
ingredient
amount
TOOL
name
cost
fuel
has

Obiekt może produkować tylko takie produkty, które są przypisane do jego typu.

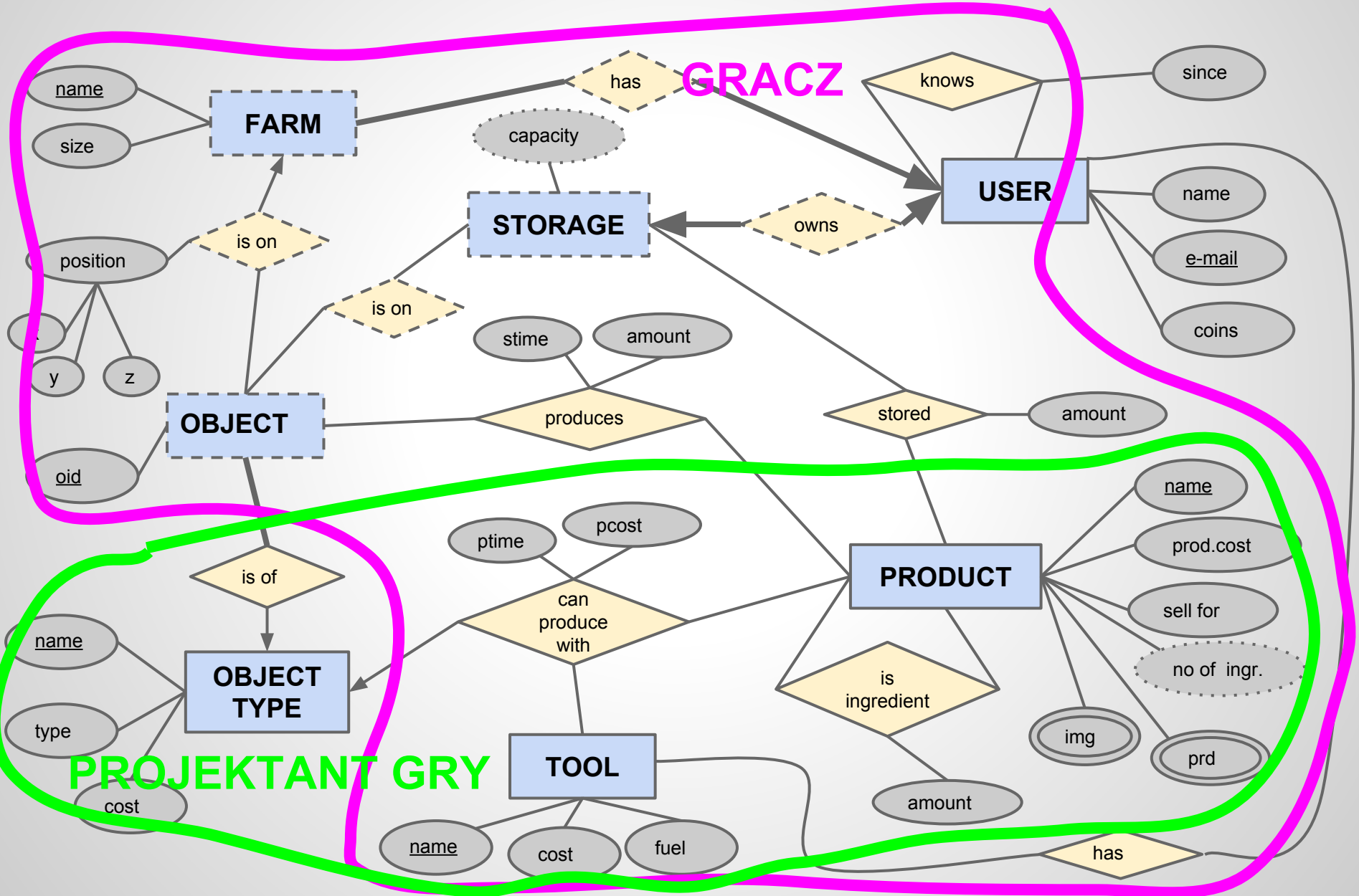



Obiekty rozmieszczone na farmie nie mogą kolidować.
name
size
FARM
has
capacity
knows
since
USER
name
e-mail
coins
STORAGE
owns
is on
position
x
y
z
is on
stime
amount
produces
OBJECT
oid
stored
amount
is of
ptime
pcost
can produce with
PRODUCT
name
prod.cost
sell for
no of ingr.
name
type
cost
OBJECT TYPE
is ingredient
img
prd
amount
TOOL
name
cost
fuel
has

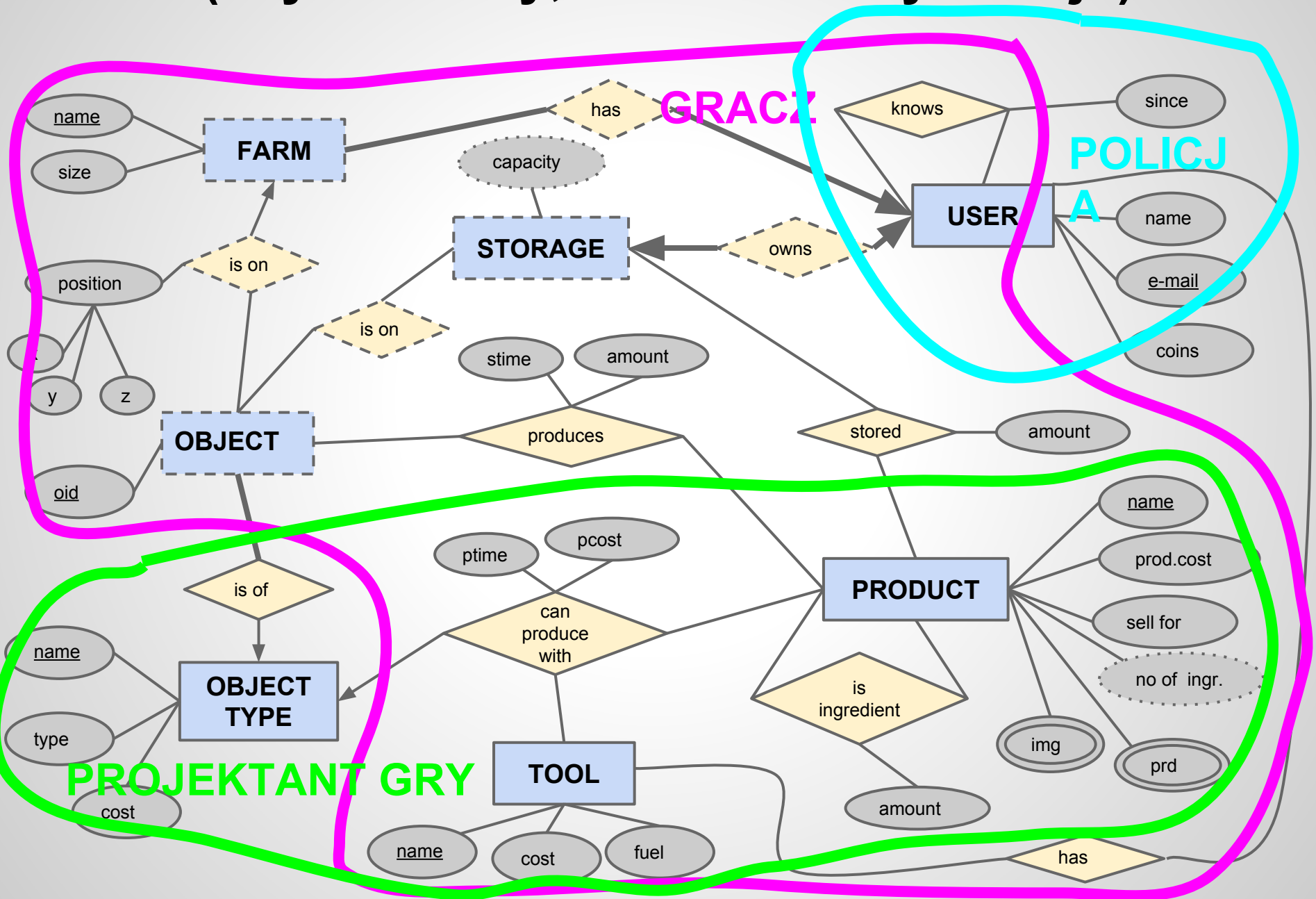
Role (użytkownicy, ich schematy i akcje)



Role (użytkownicy, ich schematy i akcje)



Role (użytkownicy, ich schematy i akcje)



Rola Gracz

- Kupuje farmy.
- Kupuje obiekty i umieszcza je na farmach. Może przesuwać obiekty. Może chować je w magazynie.
- Może uprawiać pola: orać, siać, zbierać plony, zakładając że ma na to środki (walutę). Może przy tym wykorzystywać posiadane narzędzia.
- Może uruchamiać produkcję w przedsiębiorstwach, zakładając, że ma walutę i produkty. Po zakończeniu może zebrać produkty.
- Może zbierać produkcję zwierzęcą: mleko, jajka, wełnę, zakładając, że ma odpowiednie pomieszczenie (oborę, owczarnię, kurnik).
- Może ścinać drzewa i kwiaty.
- Zebrane produkty są automatycznie dodawane do magazynu. Produkty do produkcji też są pobierane z magazynu.
- Może odwiedzać sąsiadów i u nich pracować.
- Może wysyłać wiadomości do sąsiadów.

Rola PROJEKTANT GRY

- Definiuje (nowe) rodzaje obiektów: roślin, zwierząt, produktów, przedsiębiorstw, narzędzi.
- Dla każdego obiektu określa jego etapy, czas ich trwania, obrazy obiektu na odpowiednim etapie “życia”.
- Definiuje procesy produkcji: określa składniki, potrzebne urządzenia, katalizatory.
- Może mieć dostęp do statystyk o dotychczasowym przebiegu gry: popularności jej elementów i preferencjach graczy.

Rola POLICJA

- Potrzebuje informacji o graczach i ich kontaktach.
- Może poszukiwać zachowań pasujących do konkretnego wzorca “niebezpiecznych” zachowań.
- Może poszukiwać potencjalnie niebezpiecznych zwrotów w przesyłanych wiadomościach.

Diagram E-R (dodatkowe możliwości)

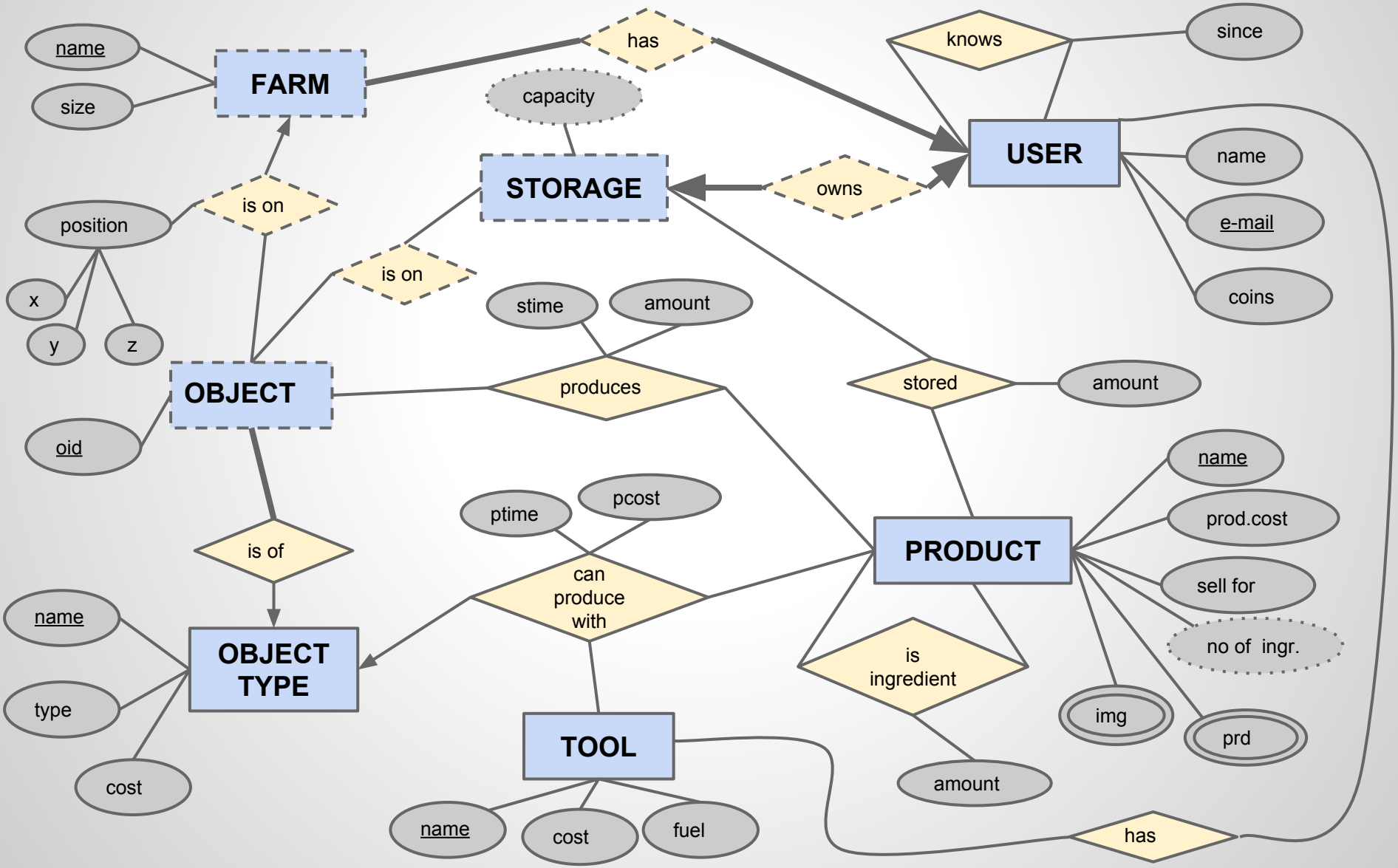


Diagram E-R (klasy i podklasy)

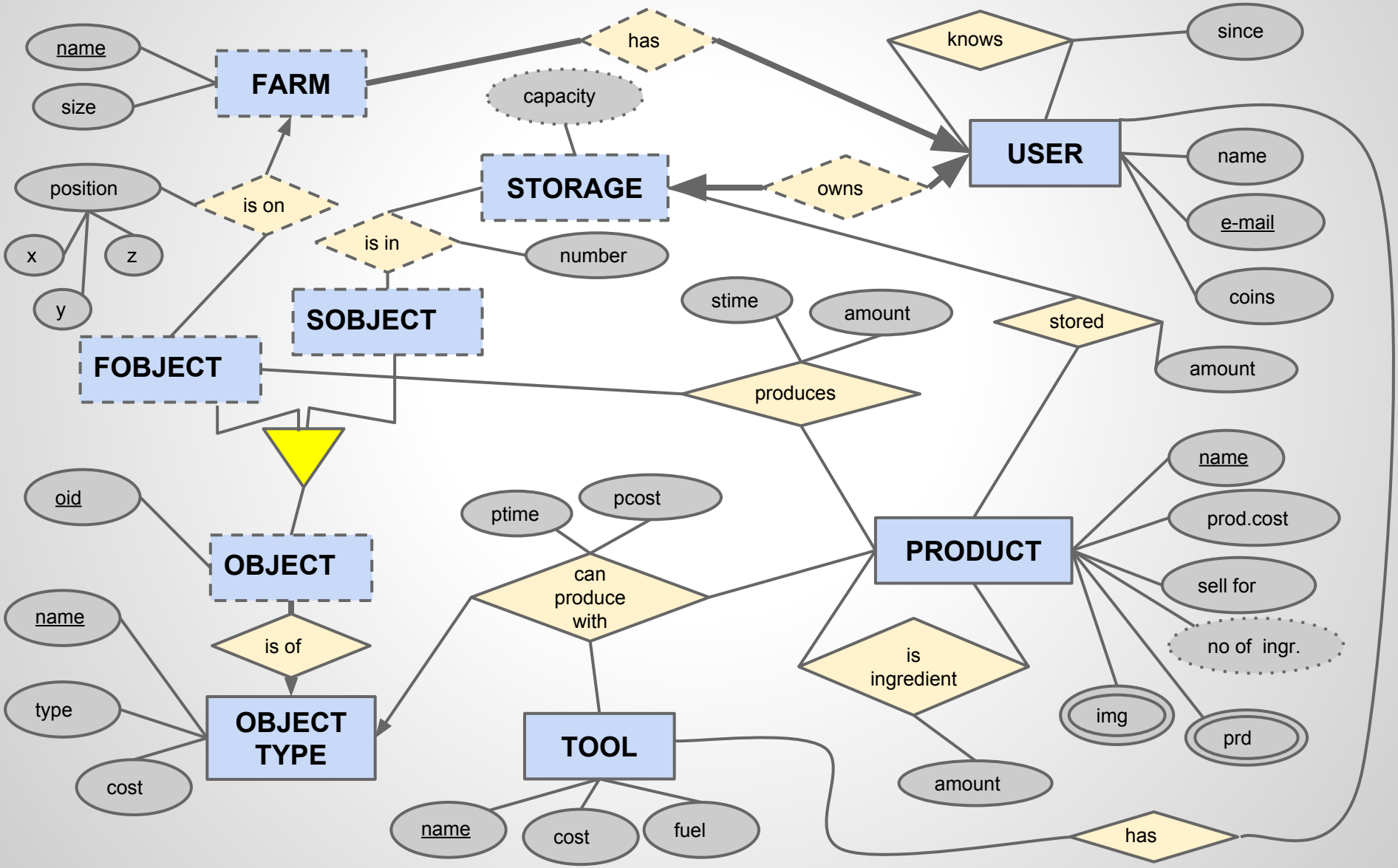


Diagram E-R (klasy i podklasy)

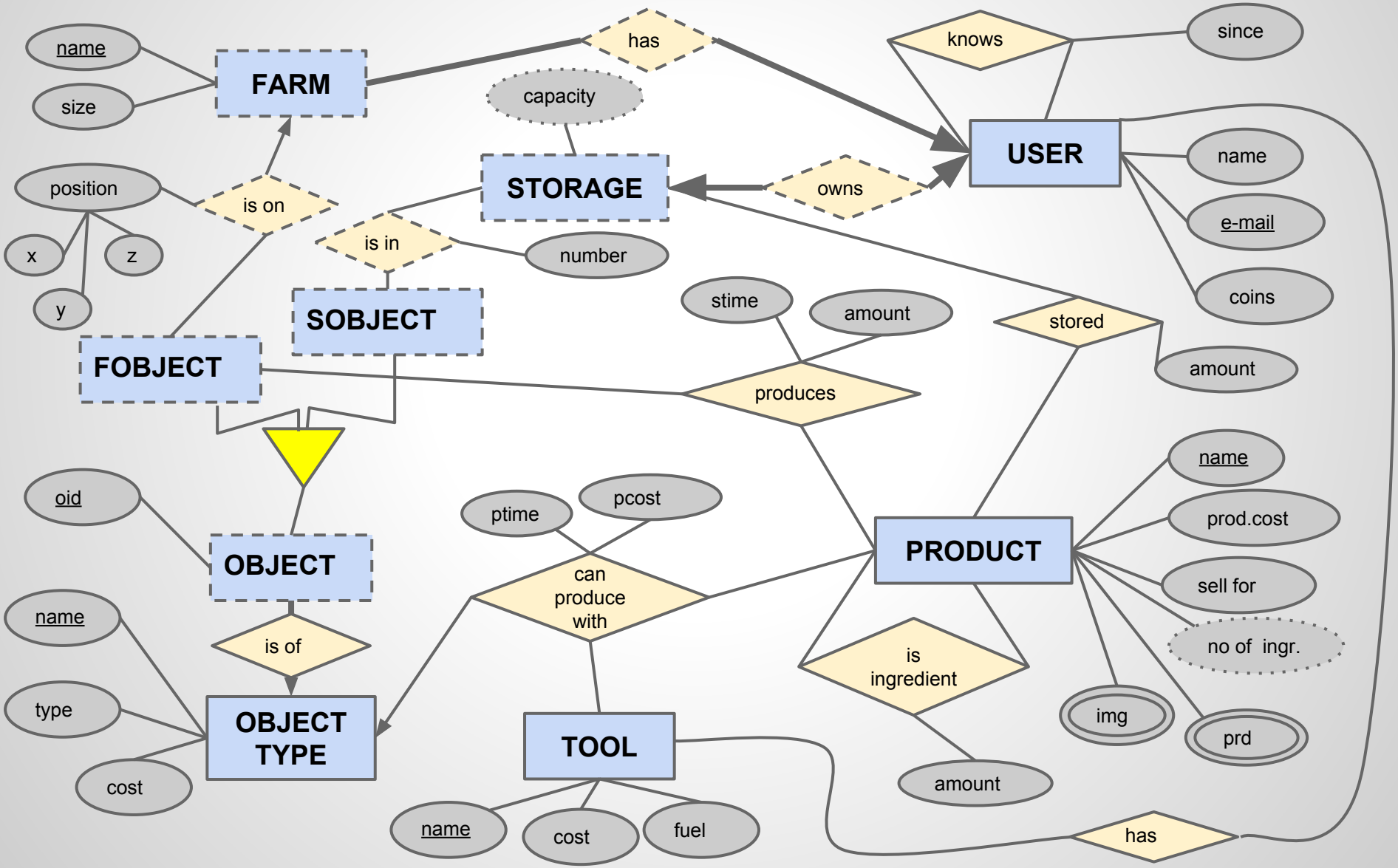


Diagram E-R (dekompozycja)

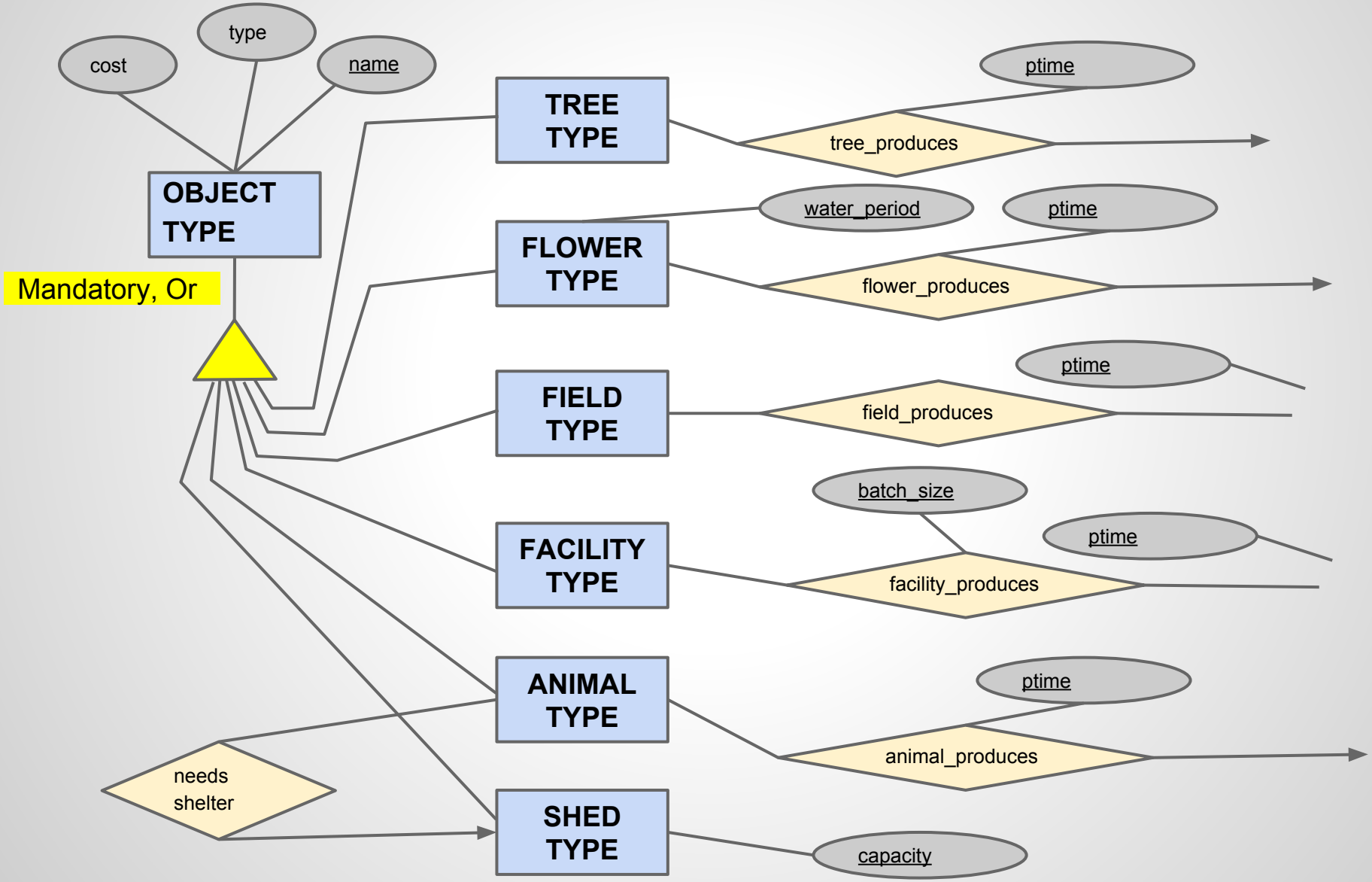
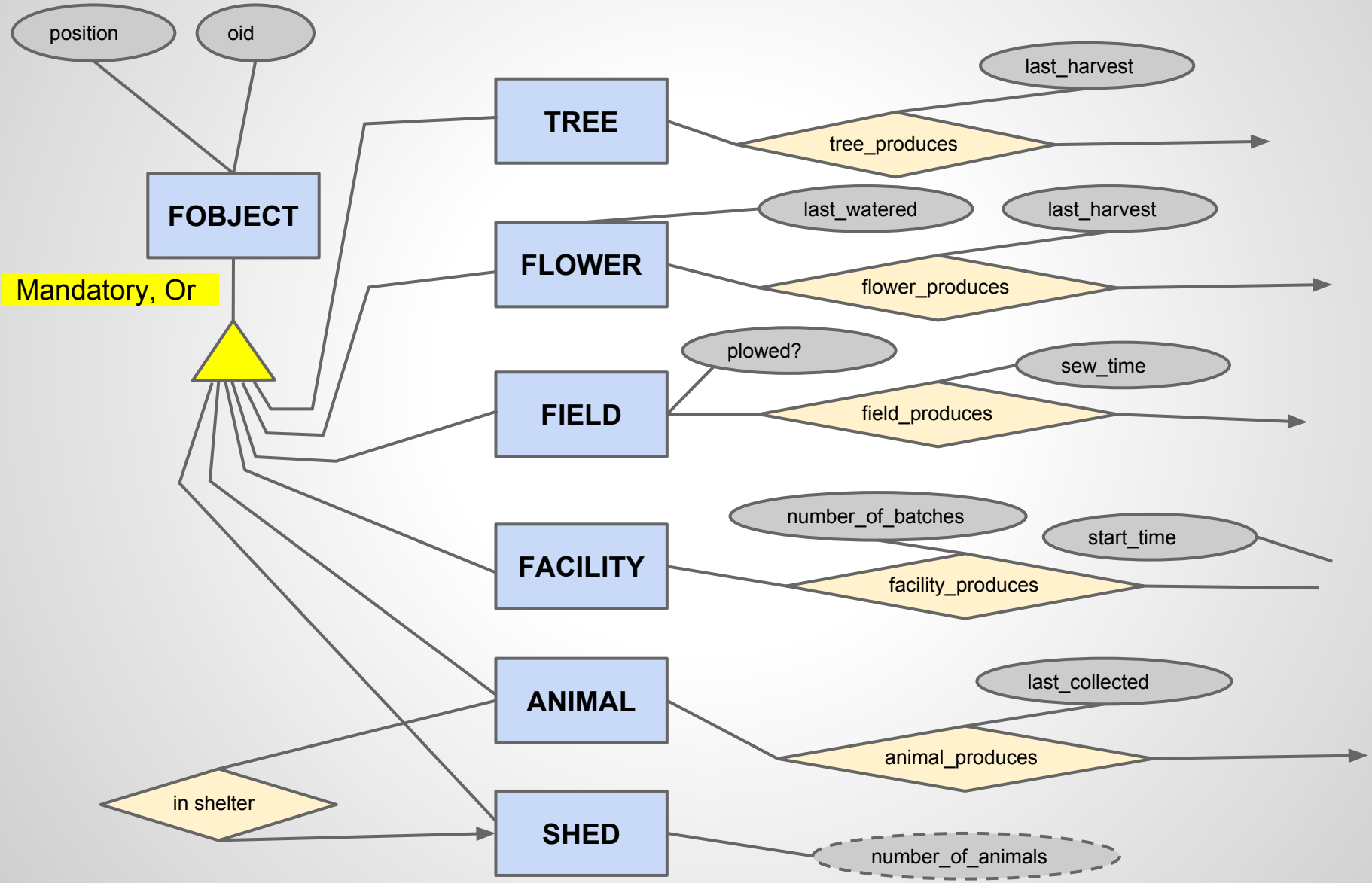


Diagram E-R (dekompozycja)



Konceptualny projekt bazy danych

- Diagram E-R (lub UML).
- Więzy nieujęte w diagramie zapisane w komentarz.
- Role wraz z podschematami i funkcjonalnościami. Dla każdej funkcjonalności trzeba wskazać obszar bazy, do którego się odwołuje.

Modelowanie fizyczne - prosty schemat

- utworzenie tabeli dla każdego **zbioru encji**; każdemu atrybutowi odpowiada kolumna w tabeli:
  - atrybuty wyliczane pomijamy
  - atrybuty wielokrotne - rezerwujemy kilka kolumn lub zapisujemy w oddzielnej tabeli;
  - dla atrybutów kluczowych nakładamy więzy i zakładamy indeksy
- zapisanie w tabelach **związków** pomiędzy zbiorami encji:
  - związek 1:n - dopisujemy klucz nadrzędnej encji i atrybuty związku do tabeli encji podrzędnej; definiujemy klucz obcy;
  - związek 1:1 - lepiej dopisać (klucz obcy) do tabeli, której udział w związku jest wymuszony;
  - związek n:m - trzeba zapisać w oddzielnej tabeli - zapisujemy w niej klucze powiązanych encji i atrybuty związku - są to klucze obce;
- **słabe zbiory encji** - tworzymy dla nich tabele i dodajemy w nich klucze encji nadrzędnej (klucze obce);
- **związki hierarchczne** - nie mieszczą się w prostym schemacie
- **więzy ogólne** - definiujemy wyzwalacze, które zachowują więzy
- **użytkownicy** - definiujemy role, uprawnienia, ewentualnie perspektywy

Model fizyczny - denormalizacja

Kontrolowana redundancja i/lub obniżenie stopnia normalizacji w celu uzyskania większej sprawności bazy (dostępu do danych).

Do tabeli `USER` dodajemy kolumny obliczane przez wyzwalacze:

- `ulevel`, `utitle`, `next_threshold` - przy każdej zmianie punktów użytkownika sprawdzamy `next_threshold`;
- `known_no` - zmiana w tabeli `KNOWS`:
  - jest ignorowana, gdy osoby już się znają
  - jest wycofywana, gdy oferujący jest zablokowany
  - jest wprowadzana (w odpowiednim porządku dla lepszej kontroli symetrii) i powoduje uaktualnienie `known_no` w tabeli `USER`.

Model fizyczny - właściwe indeksy

Ogólnego zastosowania: B-drzewo, funkcja hashująca

Specjalistyczne: R-drzewo, plik odwrócony, drzewo suffiksowe

Ogólne: wspomagają wyszukiwanie, złączenia, kontrolę unikalności:

- B-drzewo wspomaga też sortowanie i pytania z zakresu
- f.haszująca jest szybsza (~2 razy) przy prostym wyszukiwaniu

Specjalne:

- R-drzewo jest przeznaczone do przechowywania obiektów rozmieszczonych na płaszczyźnie; wspomaga przeszukiwanie obszaru, poszukiwanie obiektów w pobliżu;
- plik odwrócony pozwala przeszukiwać repozytorium tekstów w poszukiwaniu wystąpień słów, fraz, zdań;
- drzewo sufiksowe - nadaje się na indeksy dla tekstów, w których chcemy poszukiwać dowolnych podsłów (dane DNA) także z błędami.

Model fizyczny - związki hierarchiczne

ISA (agregacja): FOBJECT, GIFT ISA product-gift XOR object-gift

APO (kompozycja): STORAGE APO sobjects, products, capacity (wyliczany)

Wspólna tabela:

- dla większości obiektów na farmie możemy mieć tabelę na takie wspólne cechy, jak: aktualny obraz obiektu, kiedy ma się zmienić i na jaki, gdzie stoi obiekt - to ułatwi akcje wykonywane “samorzutnie” (samorzutna zmiana obiektu, bez ingerencji gracza);
- indywidualne cechy obiektów różnego typu mogą wymagać odrębnych tabel: pole - nawiezione ? krowa - w oborze? kiedy ostatnio wydojona?
  - kwiat - czy podlany (kiedy podlany?)

Serializacja: jedna tabela dla STORAGE z wektorem opisującym ilości posiadanych obiektów/produktów

Perspektywa: perspektywa VGIFTS pozwala pokazać listę nadesłanych prezentów z istotnymi atrybutami: obraz, ilość, nadawca

Model fizyczny - podschematy

Użytkownik GRACZ: Pokaż obraz farmy!

```
FARMS(uid,fname,fsize);
FOBJECTS(uid,fname,posX,posY,img,valid_till,nextImg);

IFARMS(uid,fname,fsize,images[][], valid_till[][],
next_img[][])
```

Użytkownik POLICJA: Szukaj aktywnych poszukiwaczy znajomości!

```
CREATE VIEW LastAcq(children, teens, adults, olds) AS ...;

CREATE INDEX MessagesWords ON MESSAGE(mtext) METHOD
<inverted_file>
```

Model fizyczny - funkcjonalności

- **Wyzwalacze:** atrybuty wyliczane, zdenormalizowane
- **Funkcje przechowywane w BD:** istotne, ale nie za bardzo skomplikowane lub nieczęsto wykonywane (efektywność!)
- **Perspektywy + reguły:** elementy podschematu
- **Aplikacja:** elementy podschematu z dużym obciążeniem obliczeniowym (np. obracanie obrazu farmy, wykrywanie kolizji w czasie umieszczania obiektu na farmie).