*Demonstration Paper*

# Using a Probabilistic Database in an Image Retrieval Application

Fajrian Yunus
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
fajrian.yunus@telecom-paris.fr

Pratik Karmakar
Department of Computer Science
National University of Singapore
Singapore
pratik.karmakar@u.nus.edu

Pierre Senellart
DI ENS, ENS, PSL University, CNRS,
Inria & IUF; CNRS@CREATE & IPAL
Paris, France & Singapore
pierre@senellart.com

Talel Abdessalem
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
talel.abdessalem@telecom-paris.fr

Stéphane Bressan*
Department of Computer Science
National University of Singapore & IPAL
Singapore
steph@comp.nus.edu.sg

## ABSTRACT

ProvSQL is a PostgreSQL extension implementing provenance management and probabilistic database features. ProvSQL seamlessly extends relational database functionality to support the storage, tracking through derivations and transformations, and querying of metadata that explain and qualify the data and query results. In this demonstration, ProvSQL is used to implement a content-based image retrieval system. A deep learning object detection model identifies objects of selected classes located within the images of a large-scale image data set. The uncertainty associated with object detection is recorded. ProvSQL's provenance model incorporates this uncertainty into the retrieval process, thus facilitating the generation of accurate and reliable results and allowing for decision-making in scenarios with incomplete or uncertain information. The demonstration illustrates how ProvSQL handles query processing, uncertainty tracking, and probability computation. It highlights the utility of a probabilistic database for applications dealing with uncertain data, compared to traditional threshold-based approaches.

## 1 INTRODUCTION

ProvSQL [17], accessible at https://provsql.org/, is an extension for PostgreSQL designed to manage the provenance of data, augmenting relational databases with advanced capabilities. It allows for the storage, tracking, and querying of metadata that document the origins, transformations, and derivations of data, providing detailed explanations and qualifications for both the data and query results. Built specifically to manage uncertain data, ProvSQL provides robust support for probabilistic databases, enabling users to model and reason about data with inherent uncertainty. In addition to tracking provenance, ProvSQL is equipped to compute the probabilities of query outputs, leveraging the uncertainty encoded in the database. Furthermore, it facilitates deeper insights into the contributions of individual database facts by computing expected Shapley values [12], a measure of their importance or impact on the (probabilistic) query results. These capabilities make ProvSQL an essential tool for applications that require both uncertainty management and explainability in data-driven decision-making processes.

In this demonstration, ProvSQL is employed to implement a content-based image retrieval system, highlighting its capabilities in managing uncertainty and delivering explainable results. The system leverages the YOLOv5 [19] object detection model to identify objects from selected classes within images from the COCO 2017 [14] dataset, a widely-used benchmark for object detection and segmentation tasks. The uncertainty inherent in the object detection process is systematically captured and recorded. ProvSQL's provenance model integrates this uncertainty to provide more granular and detailed uncertainty assessments, enhancing the reliability of the retrieval process while offering deeper insights into the confidence and likelihood of the detected objects. This approach enables the generation of more informative and dependable outcomes, supporting robust decision-making even in scenarios with incomplete or uncertain information.

The demonstration highlights how ProvSQL enables complex queries and probability computations on uncertain data. Users can retrieve results by decreasing likelihood of certain object combinations appearing in an image. This is compared with a more traditional approach, which is to set a confidence threshold on the individual objects, and then treating the resulting dataset as perfect.

We provide a short video illustrating the demonstration scenario at https://provsql.org/coco_demo_video while the code and dataset used to support the demonstration is available at https://provsql.org/coco_demo.

## 2 RELATED WORK

ProvSQL captures various provenance formalisms, including provenance semirings [10], where-provenance [5], semirings with monus [8] (an extension of provenance semirings for non-monotone queries), and provenance semimodules for aggregate queries [2]. For an overview on how these different provenance frameworks were implemented within ProvSQL, see [15, 16].

ProvSQL is the first system to support both (m-)semiring provenance and probabilistic query evaluation within a unified framework. Unlike previously developed systems like MayBMS [11], Trio [4], Orion [6], or Perm [9], which were built by modifying the internals of obsolete PostgreSQL versions, ProvSQL is easily deployable on modern PostgreSQL installations without modifying the database engine. ProvSQL shares non-probabilistic

*Stéphane Bressan tragically passed away in December 2024, while this research was conducted. This demonstration paper is dedicated to his memory.

**Table 1: Illustrating simplified excerpt of the `dataset` table**

`dataset`

| id | img | obj | prob | provenance |
|---|---|---|---|---|
| 1 | img1 | 50 | 0.5 | $t_1$ |
| 2 | img1 | 49 | 0.2 | $t_2$ |
| 3 | img1 | 45 | 0.4 | $t_3$ |
| 4 | img1 | 45 | 0.6 | $t_4$ |
| 5 | img2 | 23 | 0.8 | $t_5$ |
| 6 | img2 | 23 | 0.7 | $t_6$ |
| 7 | img3 | 75 | 0.9 | $t_7$ |
| 8 | img3 | 58 | 0.3 | $t_8$ |

features with systems like Perm and GProM [3], but uniquely integrates probabilistic data support.

## 3 PROBABILISTIC DATABASES AND PROVSQL

Let us discuss how provenance can be used for probabilistic query evaluation (Section 3.1) before discussing the way this is implemented in ProvSQL in Section 3.2.

### 3.1 Provenance and Probabilities

Provenance [5], also known as data lineage [7], refers to the extra information or metadata attached to data to trace its origins and transformations through various computational processes. It provides a detailed account of how query results in a database are derived from the input data. The concept of provenance semirings [10], developed by Green, Karvounarakis, and Tannen, offers a mathematical framework for tracking provenance. In this framework, data is annotated with elements from a semiring – an algebraic structure characterized by two binary operations (addition and multiplication) and their respective identity elements.

Consider the relation `dataset` that we use in the demonstration. It contains the following attributes: `id` (unique identifier), `img` (image identifier), `obj` (object class), `prob` (probability of object detected from the respective class), and `provenance` (provenance token). It stores object detection results for multiple images, with each row representing a detected object, its associated probability, and provenance token. Though ProvSQL can represent more complex probabilistic dependencies, in the absence of information about correlations on object annotations, we assume a simple *tuple-independent database* probabilistic model [18], where every object annotation is probabilistically independent of every other annotation. We show an illustrating simplified excerpt of this data in Table 1.

We explain what happens when we run the following query in a provenance-aware database:

```
SELECT DISTINCT a.img
FROM dataset a JOIN dataset b ON a.img = b.img
WHERE a.obj = 50 AND b.obj = 45
```

The query checks whether there exists an image in the dataset that contains at least one object of class 50 and at least one object of class 45. In the *Boolean function semiring* [15], the provenance of the query result can be computed to be the following expression:

$$\varphi_{\text{ex}} = (t_1 \wedge t_3) \vee (t_1 \wedge t_4)$$

This means that the result is derived either from the combination of evidence from $t_1$ and $t_3$, or from $t_1$ and $t_4$, where $t_1$, $t_3$, and $t_4$ are the provenance annotations from the table rows corresponding to the two object classes of interest (50 and 45).

The probability of the query result can be computed based on the provenance formula. This is in general a complex (#P-hard) problem, but for this example the computation is easy since the provenance is *read-once*:

$$\Pr(\varphi_{\text{ex}}) = 1 - (1 - \Pr(t_1) \times \Pr(t_3)) \times (1 - \Pr(t_1) \times \Pr(t_4)).$$

From Table 1, we get the values of $\Pr(t_1), \Pr(t_3)$, and $\Pr(t_4)$. Substituting these values into the formula, we get:

$$\Pr(\varphi_{\text{ex}}) = 1 - (1 - 0.5 \times 0.4) \times (1 - 0.5 \times 0.6) = 1 - 0.56 = 0.44.$$

Therefore, the probability of the query result is $\Pr(\varphi_{\text{ex}}) = 0.44$.

### 3.2 Implementation and Features of ProvSQL

ProvSQL [17] is a PostgreSQL extension that enables provenance tracking through the semiring [10] framework and extensions thereof, allowing data to be annotated and queried with provenance information. Provenance is computed in a free term algebra (which specializes to the integer polynomial semiring $\mathbb{N}[X]$ for semiring provenance), and maps database operations like selection, projection, and joins to corresponding semiring (and non-semiring) operations.

Instead of using provenance formulas, ProvSQL stores provenance computations as arithmetic circuits, reducing both storage needs and processing time. To handle SQL's multiset semantics, ProvSQL adapts the semiring operations to account for duplicates in SQL operations such as `GROUP BY`, `DISTINCT`, and `UNION`. To support operations involving negation, such as `EXCEPT`, ProvSQL leverages the m-semiring approach of [8]. Aggregate queries (e.g., `SUM`, `COUNT`, `MIN`) are managed by employing semimodules [2], allowing annotation of specific attributes.

Utilizing provenance, ProvSQL supports probabilistic query evaluation on probabilistic databases through a variety of techniques: independent computation of provenance, decomposition of the circuit when it is low-treewidth [1], or use of external knowledge compilers such as d4 [13] as a last resort. It also allows (expected) Shapley and Banzhaf value computations [12], offering insights into how much a specific data item contributes to query results.

While ProvSQL is feature-rich, certain functionalities such as recursive queries and nested aggregation remain under development.

## 4 DEMONSTRATION SCENARIO

We now present the dataset we use in our demonstration (Section 4.1) and the way the user will interact with the system (Section 4.2).

### 4.1 Dataset

Our demonstration is based on the standard COCO 2017 object detection dataset [14]. It allows the user to find images which contain certain combinations of objects. An image can contain several objects of the same type and several types of objects. Each object annotation has an independent probability value of being correct, which is stored along with the annotation in a SQL table similar to that of Table 1. The probability values in our database come from the confidence score of a state-of-the-art deep learning object detection method, YOLOv5 [19], which

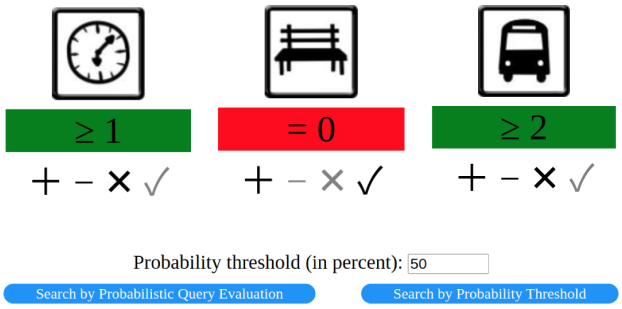



**Figure 1: User interface to run a query. Here, we search for images which contain at least one clock, at least two buses (minimum count), and no bench (absence).**

reflect its confidence in the annotation and how obvious each object is in the picture.

To demonstrate processing of low-confidence data, we enrich the dataset with *stained* version of the images in COCO 2017 dataset, using standard image filters and adding moderately irregular polygons with brown background whose opacity is set to be a random value around 50%. We also apply a box blur inside the stained area. This staining is a form of image degradation. YOLOv5 is sensitive to this degradation, and thus produces a number of low-quality, low-confidence, annotations as a result. Altogether, our dataset is formed of ≈ 240 000 images with ≈ 1 570 000 object annotations.

## 4.2 User Interface

Using a graphical interface (see Figure 1), the user can specify different queries. For each possible object annotation, the user has the possibility of fixing its minimum count (count $\geqslant n$ where $n \geqslant 1$) or requiring the object to be absent (count = 0). Minimum count queries allow the users to find images which contain at least a certain number of objects of a certain type (e.g., find images which have at least three dogs). Absence queries allow the users to find images which do not contain a certain object (e.g., find images which have no dog). Note that due to the nature of probabilistic databases, an object with the probability of 0.7 has probability of $1 - 0.7 = 0.3$ of not existing, and therefore will still appear in query results requiring the object to be absent, but with a lower probability. These two query types can be combined (e.g., find images which have at least one person and no dog).

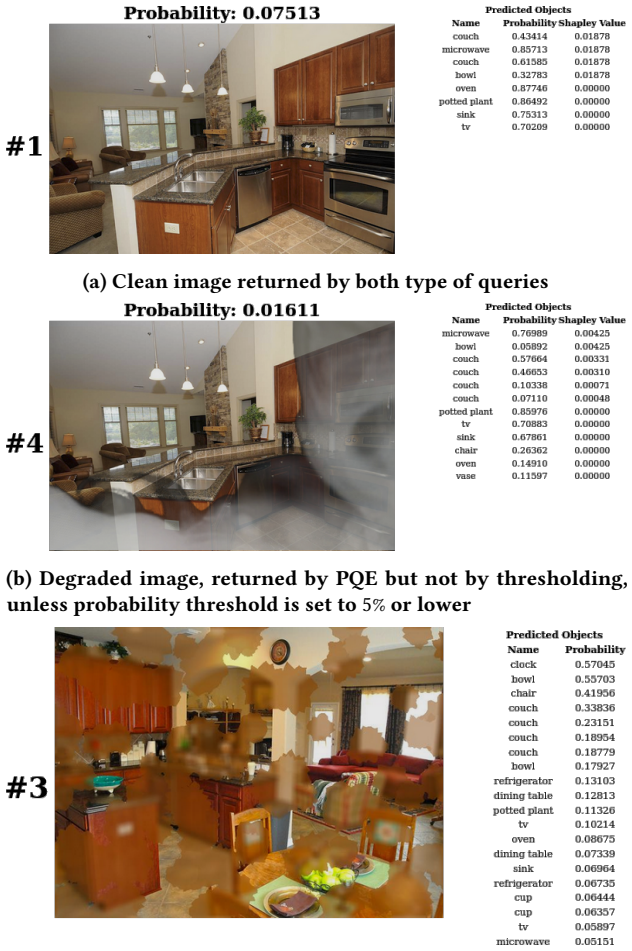
As illustrated in Figure 1, the user can choose the object(s) by clicking the + − × ✓signs which respectively stand for increasing the minimum count, decreasing it, requiring the object to be absent, allowing the object to be present. The user can then choose to search either by running the query within ProvSQL as probabilistic query evaluation, ordering query results by decreasing probabilities; or by using a non-probabilistic approach, simply setting a threshold on probabilities of annotation. This results in SQL queries being constructed and sent to PostgreSQL (extended with ProvSQL) for evaluation. For the example of Figure 1 (knowing that clock, bench, bus have classes 74, 13, 5), ne obtains respectively, for ProvSQL's probabilistic query evaluation (PQE):

```
SELECT img, probability_evaluate(provenance()) prob
FROM (
  SELECT DISTINCT p1.img
  FROM dataset c, dataset b1, dataset b2
  WHERE c.img=b1.img AND c.img=b2.img
    AND c.obj=74 AND b1.obj=5 AND b2.obj=5
    AND b1.id<>b2.id
  EXCEPT
  SELECT DISTINCT img FROM dataset
  WHERE obj=13) t
ORDER by prob DESC
```

and for thresholding (say, with threshold 50%), without any ProvSQL feature on a dataset without provenance tracking:

```
SELECT DISTINCT p1.img
FROM dataset c, dataset b1, dataset b2
WHERE c.img=b1.img AND c.img=b2.img
  AND c.obj=74 AND b1.obj=5 AND b2.obj=5
  AND b1.id<>b2.id
  AND c.prob>=.5 AND b1.prob>=.5 AND b2.prob>=.5
EXCEPT
SELECT DISTINCT img FROM dataset
WHERE obj=13 AND prob>=.5
```

After the user runs a probabilistic query evaluation search, he or she will get the raw number of result tuples, the expected value of this number computed using ProvSQL's probabilistic evaluation of aggregated queries, and each resulting image. For




**(a) Clean image returned by both type of queries**

**(b) Degraded image, returned by PQE but not by thresholding, unless probability threshold is set to 5% or lower**

**(c) Image that does not satisfy the query, but which is returned by thresholding if probability threshold is set to 5% or lower**

**Figure 2: Example case where PQE return valid images that are impossible to separate from invalid ones through the thresholding approach**

each returned image, the probability that the image satisfies the query is displayed, along with all predicted objects inside the image with their respective probability and their expected Shapley value as a measure of how much they contribute to the result. Finally, we also let users issue free-form SQL queries on the database, to better understand how probabilistic query evaluation works in ProvSQL.

As an example of case illustrating the value of probabilistic query evaluation over thresholding, consider a query where a user looks for a picture with $\geqslant 2$ couches, $\geqslant 1$ bowl, and $\geqslant 1$ microwave oven. An example of such image is obtained in Figure 2a, which is the top result for PQE, and also obtained when using the thresholding method as long as the threshold is $\leqslant 32\%$. Now, a degraded version of the same image (Figure 2b) also matches, with lower confidence score for PQE (it is ranked #4, after two other valid matches); but to obtain this result with thresholding, one needs to set the threshold as low as 5%. But then, with such a low threshold, the thresholding method also returns wrong results, such as Figure 2c where a TV is wrongly labeled as a microwave. In addition to the difficulty of finding the right threshold, it is common that the thresholding approach cannot separate right results from wrong, but where probabilistic query evaluation, by taking into account the combined confidence in every annotation, correctly ranks images.

## ACKNOWLEDGMENTS

This research is part of the DesCartes program and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program. This work is made in memory of our co-author, colleague, and friend Stéphane Bressan, who unexpectedly passed away in December 2024.

## REFERENCES

[1] Antoine Amarilli, Florent Capelli, Mikaël Monet, and Pierre Senellart. 2018. Connecting Knowledge Compilation Classes and Width Parameters. *CoRR* abs/1811.02944 (2018). arXiv:1811.02944 http://arxiv.org/abs/1811.02944

[2] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for aggregate queries. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, Maurizio Lenzerini and Thomas Schwentick (Eds.). ACM, 153–164. https://doi.org/10.1145/1989284.1989302

[3] Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. 2018. GProM - A Swiss Army Knife for Your Provenance Needs. *IEEE Data Eng. Bull.* 41, 1 (2018), 51–62. http://sites.computer.org/debull/A18mar/p51.pdf

[4] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. 2006. ULDBs: Databases with Uncertainty and Lineage. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim (Eds.). ACM, 953–964. http://dl.acm.org/citation.cfm?id=1164209

[5] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings (Lecture Notes in Computer Science)*, Jan Van den Bussche and Victor Vianu (Eds.), Vol. 1973. Springer, 316–330. https://doi.org/10.1007/3-540-44503-X_20

[6] Reynold Cheng, Sarvjeet Singh, and Sunil Prabhakar. 2005. U-DBMS: A Database System for Managing Constantly-Evolving Data. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi (Eds.). ACM, 1271–1274. http://www.vldb.org/conf/2005/papers/p1271-cheng.pdf

[7] Yingwei Cui and Jennifer Widom. 2000. Practical Lineage Tracing in Data Warehouses. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000*, David B. Lomet and Gerhard Weikum (Eds.). IEEE Computer Society, 367–378. https://doi.org/10.1109/ICDE.2000.839437

[8] Floris Geerts and Antonella Poggi. 2010. On database query languages for K-relations. *J. Appl. Log.* 8, 2 (2010), 173–185. https://doi.org/10.1016/J.JAL.2009.09.001

[9] Boris Glavic and Gustavo Alonso. 2009. Perm: Processing Provenance and Data on the Same Data Model through Query Rewriting. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng (Eds.). IEEE Computer Society, 174–185. https://doi.org/10.1109/ICDE.2009.15

[10] Todd J. Green, Gregory Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, Leonid Libkin (Ed.). ACM, 31–40. https://doi.org/10.1145/1265530.1265535

[11] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. 2009. MayBMS: a probabilistic database management system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul (Eds.). ACM, 1071–1074. https://doi.org/10.1145/1559845.1559984

[12] Pratik Karmakar, Mikaël Monet, Pierre Senellart, and Stéphane Bressan. 2024. Expected Shapley-like scores of boolean functions: Complexity and applications to probabilistic databases. *Proceedings of the ACM on Management of Data* 2, 2 (2024), 1–26.

[13] Jean-Marie Lagniez and Pierre Marquis. 2017. An Improved Decision-DNNF Compiler. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 667–673. https://doi.org/10.24963/IJCAI.2017/93

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V (Lecture Notes in Computer Science)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.), Vol. 8693. Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[15] Pierre Senellart. 2017. Provenance and Probabilities in Relational Databases. *SIGMOD Rec.* 46, 4 (2017), 5–15. https://doi.org/10.1145/3186549.3186551

[16] Pierre Senellart. 2024. On the Impact of Provenance Semiring Theory on the Design of a Provenance-Aware Database System. In *The Provenance of Elegance in Computation - Essays Dedicated to Val Tannen, Tannen's Festschrift, May 24-25, 2024, University of Pennsylvania, Philadelphia, PA, USA (OASIcs)*, Antoine Amarilli and Alin Deutsch (Eds.), Vol. 119. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:10. https://doi.org/10.4230/OASICS.TANNEN.9

[17] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. 2018. ProvSQL: Provenance and Probability Management in PostgreSQL. *Proc. VLDB Endow.* 11, 12 (2018), 2034–2037. https://doi.org/10.14778/3229863.3236253

[18] Dan Suciu. 2020. Probabilistic Databases for All. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, Dan Suciu, Yufei Tao, and Zhewei Wei (Eds.). ACM, 19–31. https://doi.org/10.1145/3375395.3389129

[19] Ultralytics. 2021. YOLOv5: A state-of-the-art real-time object detection system. https://docs.ultralytics.com.